EXHIBIT A

US010324702B2

(12) **United States Patent**
Vishnepolsky et al.

(10) **Patent No.: US 10,324,702 B2**
(45) **Date of Patent: Jun. 18, 2019**

(54) **CLOUD SUFFIX PROXY AND A METHOD THEREOF**

(71) Applicant: **Adallom Technologies Ltd.**, Tel Aviv (IL)

(72) Inventors: **Gregory Vishnepolsky**, Rehovot (IL); **Liran Moysi**, Ramat Gan (IL)

(73) Assignee: **MICROSOFT ISRAEL RESEARCH AND DEVELOPMENT (2002) LTD.**, Matam Haifa (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 483 days.

(21) Appl. No.: **14/847,469**

(22) Filed: **Sep. 8, 2015**

(65) **Prior Publication Data**

US 2016/0077824 A1 Mar. 17, 2016

Related U.S. Application Data

(63) Continuation-in-part of application No. 14/539,980, filed on Nov. 12, 2014, now Pat. No. 9,438,565.
(Continued)

(51) **Int. Cl.**

G06F 9/44	(2018.01)
G06F 8/65	(2018.01)

(Continued)

(52) **U.S. Cl.**
CPC ***G06F 8/65*** (2013.01); ***G06F 16/958*** (2019.01); ***G06F 17/2247*** (2013.01);
(Continued)

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,397,246	B1	5/2002	Wolfe
7,571,217	B1	8/2009	Saxena

(Continued)

FOREIGN PATENT DOCUMENTS

JP	2006526843	A	11/2006
JP	2009289206	A	12/2009

(Continued)

OTHER PUBLICATIONS

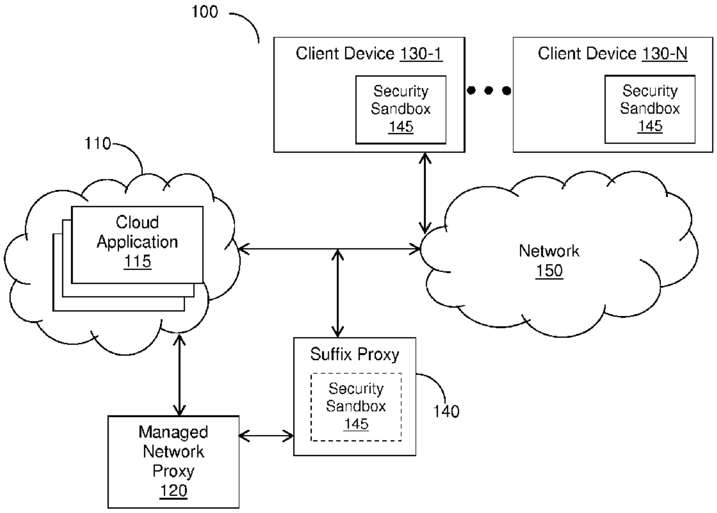
"Final Office Action Issued in U.S. Appl. No. 14/539,980", dated Dec. 8, 2015, 13 Pages.
(Continued)

Primary Examiner — Insun Kang
(74) *Attorney, Agent, or Firm* — M&B IP Analysts, LLC.

(57) **ABSTRACT**

A method and system for modifying network addresses of at least one cloud application. The method comprises receiving a webpage sent to a client device from the at least one cloud application, wherein a webpage designates at least one script loaded to the client device during runtime; injecting a piece of code to the webpage; receiving, by the injected piece of code, an attempt to load each of the at least one script; modifying the at least one script by suffixing each network address designated in the at least one script with a predefined network address; and sending the modified at least one script to the client device, wherein runtime execution of the modified at least one script on the client device causes redirection of future requests from the client device to the cloud application to the suffixed network address.

33 Claims, 4 Drawing Sheets



Related U.S. Application Data

(60) Provisional application No. 62/049,473, filed on Sep. 12, 2014.

(51) **Int. Cl.**

G06F 21/53	(2013.01)
G06F 17/22	(2006.01)
G06F 16/958	(2019.01)
H04L 29/08	(2006.01)
H04L 29/12	(2006.01)

(52) **U.S. Cl.**
CPC ***G06F 21/53*** (2013.01); ***H04L 67/02*** (2013.01); ***H04L 67/1002*** (2013.01); ***H04L 67/2814*** (2013.01); *H04L 61/301* (2013.01); *H04L 61/3055* (2013.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,873,707	B1 *	1/2011	Subramanian	G06F 17/30887 709/219
7,958,232	B1	6/2011	Colton et al.	
8,543,726	B1 *	9/2013	Kann	H04L 63/0281 709/219
8,856,869	B1	10/2014	Brinskelle	
8,938,800	B2	1/2015	Bhargava et al.	
9,137,131	B1	9/2015	Sarukkai et al.	
9,154,479	B1	10/2015	Sethi	
9,369,437	B2 *	6/2016	Holloway	H04L 63/1416
9,391,832	B1 *	7/2016	Song	H04L 29/14
9,438,625	B1 *	9/2016	Yang	H04L 63/1441
2006/0117388	A1	6/2006	Nelson et al.	
2007/0016949	A1 *	1/2007	Dunagan	G06F 21/51 726/22
2007/0180510	A1	8/2007	Long et al.	
2008/0177691	A1	7/2008	Alperovitch et al.	
2009/0077369	A1 *	3/2009	Fujimaki	G06F 17/212 713/100
2009/0083726	A1 *	3/2009	Amend	H04L 63/105 717/171
2009/0217354	A1	8/2009	Blum et al.	
2009/0262741	A1	10/2009	Jungck et al.	
2009/0289206	A1	11/2009	Ding et al.	
2010/0103837	A1	4/2010	Jungck et al.	
2010/0146260	A1	6/2010	Levow et al.	
2010/0162346	A1	6/2010	Holostov et al.	
2010/0313183	A1 *	12/2010	Ellen	G06Q 30/02 717/110
2010/0325357	A1	12/2010	Reddy et al.	
2011/0029613	A1	2/2011	Hedditch	
2011/0131478	A1 *	6/2011	Tock	H04L 63/0823 715/208
2011/0208838	A1	8/2011	Thomas et al.	
2011/0257810	A1	10/2011	Leger	
2011/0289434	A1	11/2011	Kieft	
2011/0289588	A1	11/2011	Sahai et al.	
2012/0023160	A1	1/2012	Marmor	
2012/0030294	A1 *	2/2012	Piernot	G06F 17/30902 709/206
2012/0036576	A1 *	2/2012	Iyer	G06F 21/554 726/23
2012/0116896	A1 *	5/2012	Holloway	H04L 61/1511 705/14.73
2012/0137210	A1 *	5/2012	Dillon	G06F 17/30902 715/234
2012/0174236	A1 *	7/2012	Goodwin	H04L 63/102 726/27
2012/0278872	A1	11/2012	Woelfel et al.	
2013/0073609	A1 *	3/2013	Connolly	G06F 15/16 709/203
2013/0097706	A1	4/2013	Titonis et al.	
2013/0097711	A1	4/2013	Basavapatna et al.	
2013/0196396	A1	8/2013	Beeson et al.	
2013/0212689	A1 *	8/2013	Ben-Natan	H04L 63/20 726/26
2013/0276136	A1 *	10/2013	Goodwin	H04L 67/22 726/27
2013/0311863	A1 *	11/2013	Gutkin	G06F 17/30887 715/208
2013/0347094	A1	12/2013	Bettini et al.	
2014/0020072	A1	1/2014	Thomas	
2014/0032759	A1	1/2014	Barton et al.	
2014/0109175	A1	4/2014	Barton et al.	
2014/0137273	A1	5/2014	Workman	
2014/0173415	A1 *	6/2014	Kattil Cherian ..	G06F 17/30861 715/234
2014/0201524	A1	7/2014	Dittrich	
2014/0237545	A1	8/2014	Mylavarapu et al.	
2014/0282464	A1 *	9/2014	El-Gillani	G06F 8/61 717/168
2014/0282518	A1	9/2014	Banerjee	
2014/0283000	A1	9/2014	Radhakrishnan	
2014/0344332	A1 *	11/2014	Giebler	H04L 67/2823 709/203
2015/0066575	A1	3/2015	Baikalov et al.	
2015/0088968	A1 *	3/2015	Wei	H04L 67/10 709/203
2016/0119344	A1 *	4/2016	Freitas Fortuna dos Santos	H04L 63/1466 726/7
2016/0330172	A1	11/2016	Muttik	
2017/0116349	A1 *	4/2017	Steiner	G06F 17/30902
2017/0163675	A1 *	6/2017	Warman	H04L 63/1425

FOREIGN PATENT DOCUMENTS

JP	2011511974	A	4/2011
JP	2011257810	A	12/2011
JP	2013196396	A	9/2013
WO	2011094807	A1	8/2011
WO	2012063282	A1	5/2012
WO	2013091709	A1	6/2013

OTHER PUBLICATIONS

"International Search Report and Written Opinion Issued in PCT Application No. PCT/US2015/049606", dated Feb. 25, 2016, 7 Pages.
"Non-Final Office Action Issued in U.S. Appl. No. 14/539,980", dated Jan. 26, 2015, 9 Pages.
"Notice of Allowance Issued in U.S. Appl. No. 14/539,980", dated May 16, 2016, 4 Pages.
"Notice of Allowance Issued in U.S. Appl. No. 14/539,980", dated May 4, 2016, 12 Pages.
Magazinius, et al., "Architectures for Inlining Security Monitors in Web Applications", In Proceedings of 6th International Symposium on Engineering Secure Software and Systems, Feb. 26, 2014, 18 Pages.
"Non Final Office Action Issued in U.S. Appl. No. 14/968,432", dated Sep. 22, 2016, 17 Pages.
"Supplementary Search Report Issued in European Patent Application No. 14860194.1", dated May 31, 2017, 7 Pages.
"Non-Final Office Action Issued in U.S. Appl. No. 14/968,432", dated Jul. 28, 2017, 18 Pages.
Patent Cooperation Treaty International Search Report and Written Opinion for PCT/US2014/065305, ISA/US, Alexandria, VA., dated Mar. 3, 2015.
"Final Office Action Issued in U.S. Appl. No. 14/968,432", dated Mar. 23, 2017, 21 Pages.
"Office Action Issued in Australia Patent Application No. 2014346390", dated Mar. 15, 2018, 3 Pages.
"Office Action Issued in Colombian Patent Application No. 16-139041", dated Nov. 24, 2017, 17 Pages.
"Office Action Issued in Chile Patent Application No. 1116-2016", dated Jan. 26, 2018, 9 Pages.
"Office Action Issued in Mexican Patent Application No. MX/a/2016/006109", dated Nov. 29, 2017, 2 pages. (W/o English Translation).

(56) **References Cited**

OTHER PUBLICATIONS

"Non Final Office Action Issued in U.S. Appl. No. 14/968,432", dated Feb. 5, 2018, 19 Pages.
"Office Action Issued in Japanese Patent Application No. 2016-530954", dated Sep. 28, 2018, 4 Pages.
"Office Action Issued in Chile Patent Application No. 1116-2016", dated Jul. 3, 2018, 12 Pages.
"Office Action Issued in Russian Patent Application No. 2016117971", dated Jul. 9, 2018, 5 Pages.
"Office Action Issued in Chinese Patent Application No. 201480061802.1", dated Aug. 3, 2018, 11 Pages.
"Second Office Action Issued in Chinese Patent Application No. 201480061802.1", dated Mar. 14, 2019, 7 Pages.
"Supplementary Examination Report Issued in Singapore Patent Application No. 11201603471X", dated Mar. 20, 2019, 3 Pages.

* cited by examiner

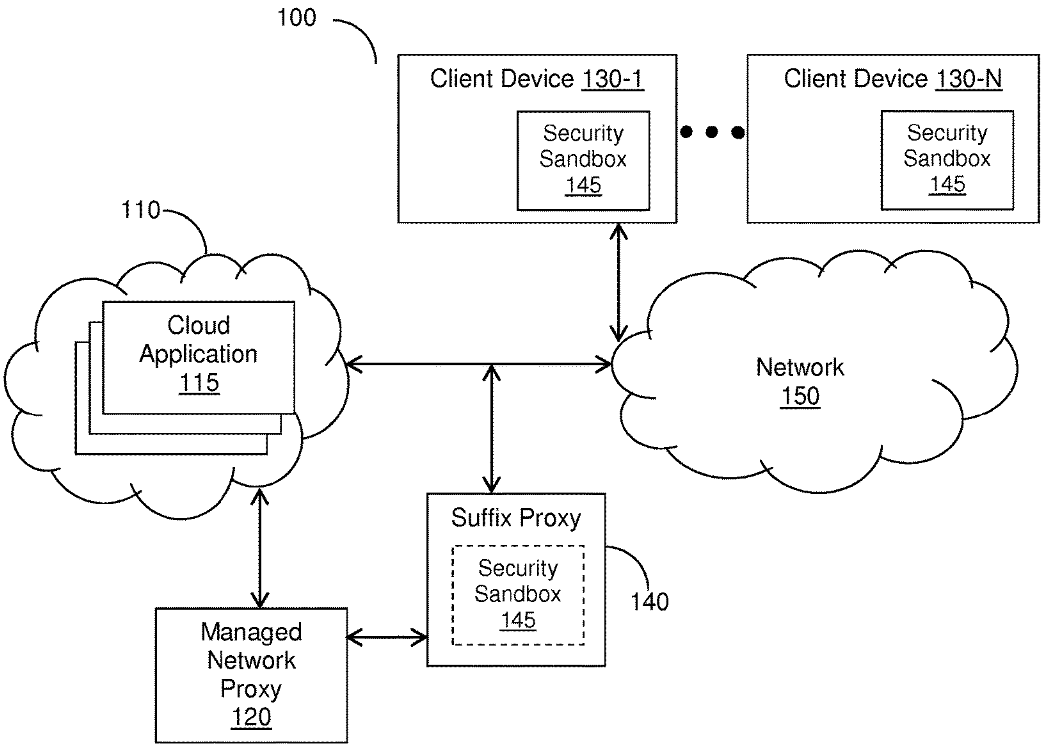

100
Client Device 130-1
Security Sandbox 145
Client Device 130-N
Security Sandbox 145
110
Cloud Application 115
Network 150
Suffix Proxy
Security Sandbox 145
140
Managed Network Proxy 120

FIG. 1

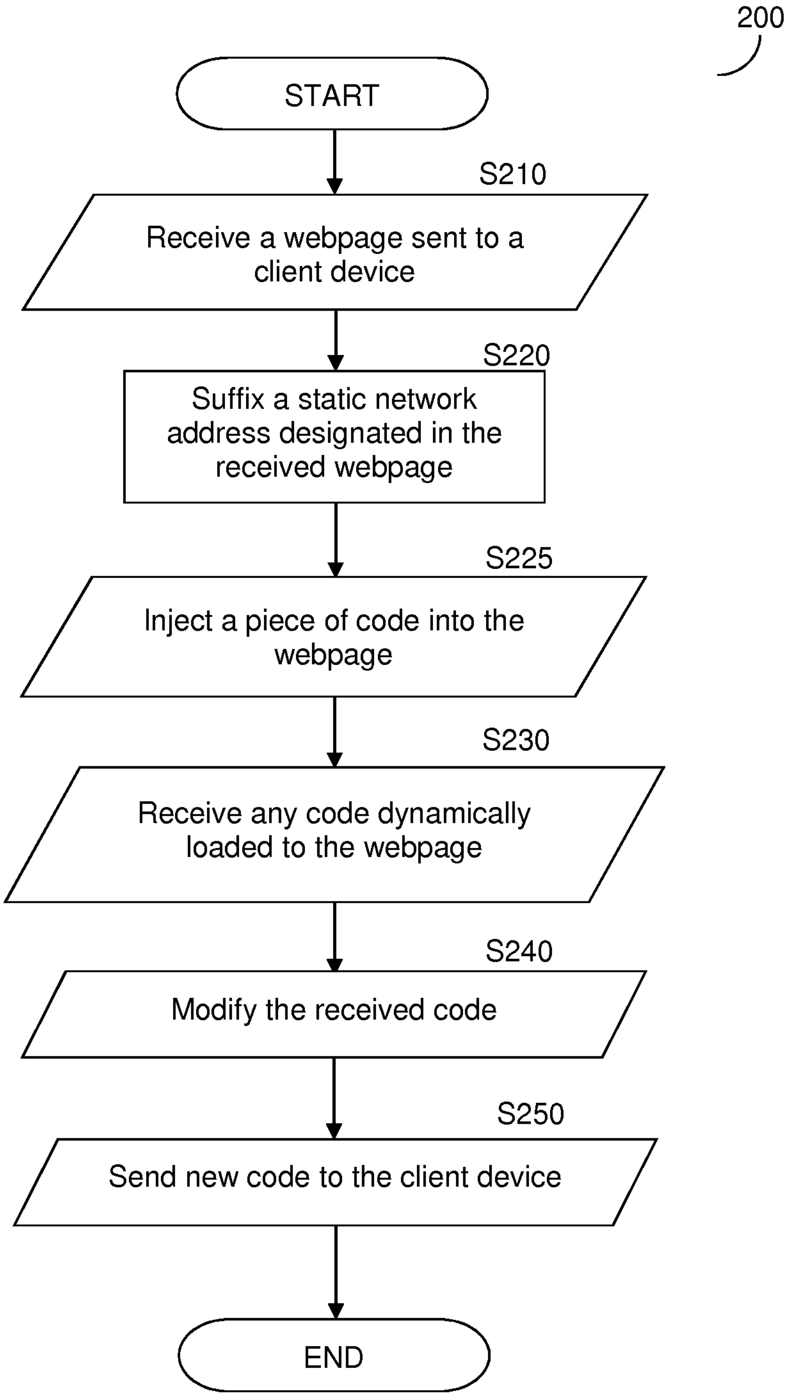

200
START
S210
Receive a webpage sent to a client device
S220
Suffix a static network address designated in the received webpage
S225
Inject a piece of code into the webpage
S230
Receive any code dynamically loaded to the webpage
S240
Modify the received code
S250
Send new code to the client device
END

FIG. 2

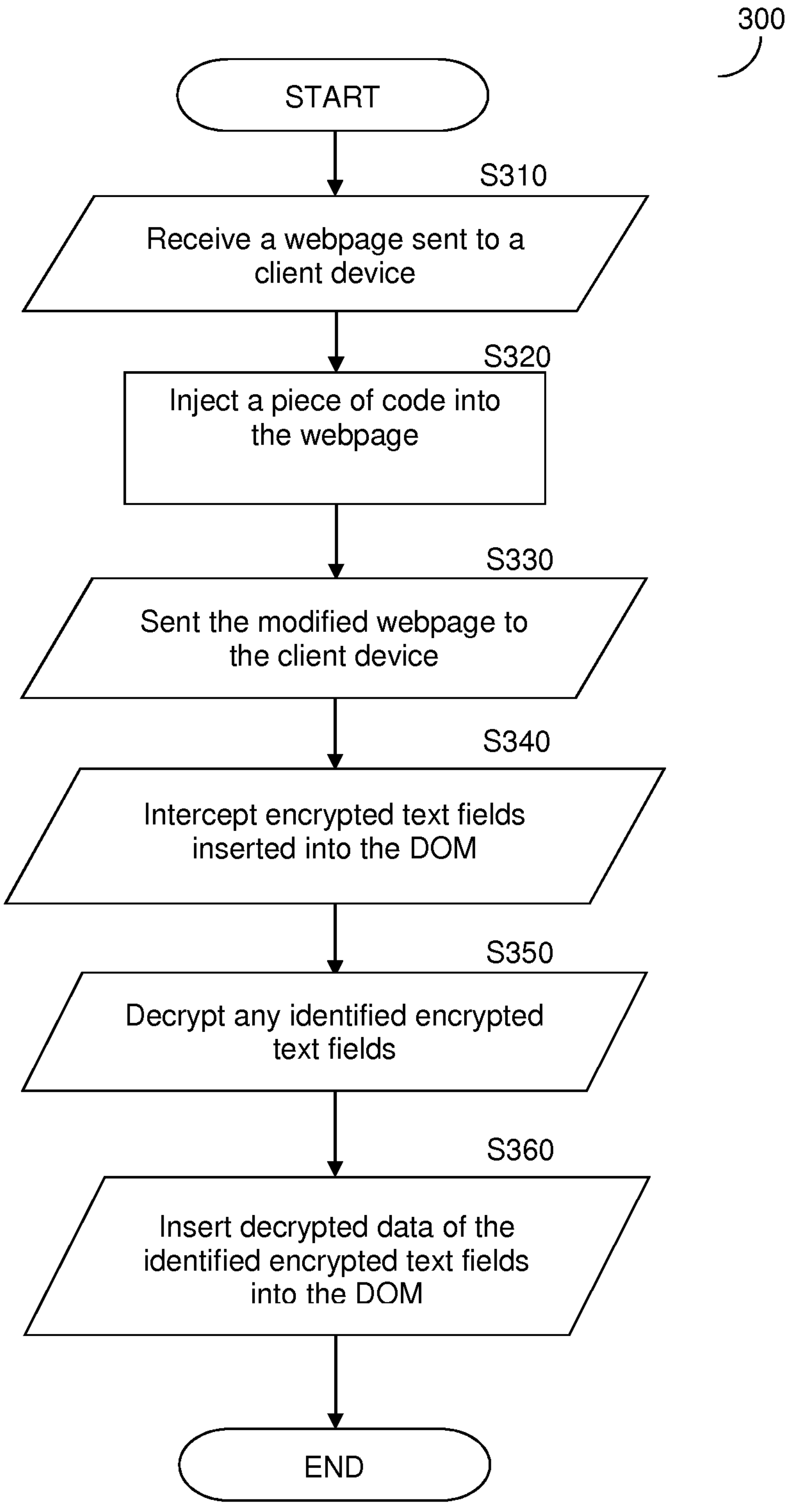

300
START
S310
Receive a webpage sent to a client device
S320
Inject a piece of code into the webpage
S330
Sent the modified webpage to the client device
S340
Intercept encrypted text fields inserted into the DOM
S350
Decrypt any identified encrypted text fields
S360
Insert decrypted data of the identified encrypted text fields into the DOM
END

FIG. 3

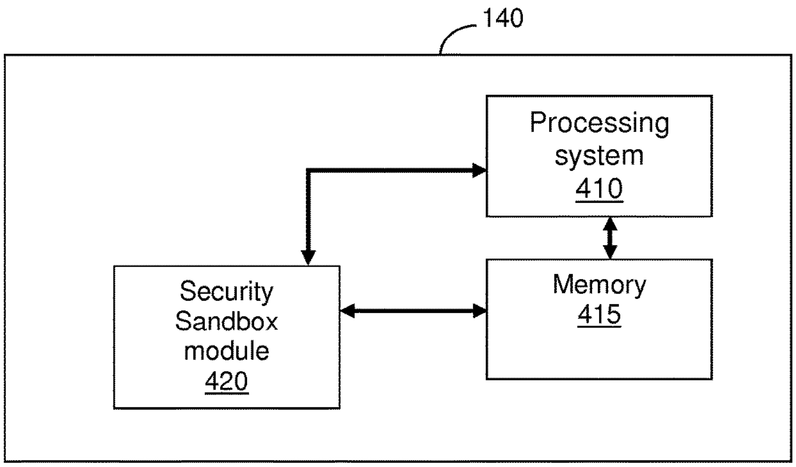

140
Processing system 410
Memory 415
Security Sandbox module 420

Fig. 4

CLOUD SUFFIX PROXY AND A METHOD THEREOF

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 62/049,473 filed on Sep. 12, 2014. This application is also a continuation-in-part of U.S. patent application Ser. No. 14/539,980 filed on Nov. 12, 2014, the contents of which are hereby incorporated by reference.

TECHNICAL FIELD

This application relates generally to securing communications networks and systems by monitoring and securing communications, in particular by use of a suffix proxy.

BACKGROUND

In recent years more and more providers offer the ability to create computing environments in the cloud. For example, Amazon Web Services™ (also known as AWS) launched in 2006 a service that provides users with the ability to configure an entire environment tailored to an application executed over a cloud platform. In general, such services allow for developing scalable applications in which computing resources are utilized to support efficient execution of the application.

Organizations and businesses that develop, provide, or otherwise maintain cloud-based applications have become accustomed to rely on these services and implement various types of environments from complex websites to applications and services provided as a software-as-service (SaaS) delivery model. Such services and applications are collectively referred to as “cloud applications.”

Cloud applications are typically accessed by users using a client device via a web browser. Cloud applications include, among others, e-commerce applications, social media applications, enterprise applications, gaming applications, media sharing applications, storage applications, software development applications, and so on. Many individual users, businesses, and enterprises turn to cloud applications in lieu of “traditional” software applications that are locally installed and managed. For example, an enterprise can use Office® 365 online services for email accounts, rather than having an Exchange® Server maintained by the enterprise.

Enterprises are increasingly adopting cloud-based SaaS offerings. These services are subject to varied network security risks. Known systems for securing these networks operate by inspecting traffic between servers operating the SaaS and the endpoint operated by a user. These known network security systems typically require complex configuration of the endpoint which increases system complexity.

Furthermore, in many cases, the endpoint may not be under the complete control of the enterprise, may be entirely unmanaged, or otherwise unconfigurable. In addition to the difficulties inherent in configuring and administering a user-controlled endpoint, it is difficult to ensure traffic captivation for an entire session when network addresses are generated dynamically.

In addition, modern web/cloud applications, such as the Google® Apps platform, utilize a large amount of client side code (JavaScript). This can make a suffix proxy implementation much more challenging, as basic proxy functions are insufficient and further intervention in the client side code is required.

It would therefore be advantageous to provide a solution that would overcome the deficiencies of the prior art techniques for capture and reconstruction of HTTP traffic.

SUMMARY

A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor delineate the scope of any or all embodiments. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the term some embodiments may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

Some embodiments of the disclosure relate to a method for modifying network addresses of at least one cloud application. The method comprises receiving a webpage sent to a client device from the at least one cloud application, wherein a webpage designates at least one script loaded to the client device during runtime; injecting a piece of code to the webpage; receiving, by the injected piece of code, an attempt to load each of the at least one script; modifying the at least one script by suffixing each network address designated in the at least one script with a predefined network address; and sending the modified at least one script to the client device, wherein runtime execution of the modified at least one script on the client device causes redirection of future requests from the client device to the cloud application to the suffixed network address.

Some embodiments of the disclosure relate to a system for modifying network addresses of at least one cloud application. The system comprises a processor; and a memory containing instructions that, when executed by the processor, configure the system to: receive a webpage sent to a client device from the at least one cloud application, wherein a webpage designates at least one script loaded to the client device during runtime; inject a piece of code to the webpage; receive, by the injected piece of code, an attempt to load each of the at least one script; modify the at least one script by suffixing each network address designated in the at least one script with a predefined network address; and send the modified at least one script to the client device, wherein runtime execution of the modified at least one script on the client device causes redirection of future requests from the client device to the cloud application to the suffixed network address.

BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

FIG. **1** is a diagram of a networked system utilized to describe the various disclosed embodiments.

FIG. **2** is a flowchart illustrating the operation of the security sandbox according to one embodiment.

FIG. **3** is a flowchart illustrating a method for controlling changes to the DOM according to one embodiment.

FIG. **4** is a block diagram of a suffix proxy implemented according to an embodiment.

DETAILED DESCRIPTION

It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

By a way of example, the various disclosed embodiments can be configured to operate on network traffic between a network-based software as a service (SaaS) provider and a client. As will be discussed in greater detail below, the disclosed embodiments allow for non-intrusive suffixing and un-suffixing of network addresses directed to the SaaS provider.

FIG. **1** is an exemplary and non-limiting diagram of a networked system **100** utilized to describe the various disclosed embodiments. The networked system **100** includes a cloud computing platform **110** which may be a private cloud, a public cloud, or a hybrid cloud providing computing resources to applications or services executed therein. In an embodiment, the cloud computing platform **110** may be of a SaaS platform.

Organizations and businesses that develop, provide, or otherwise maintain cloud based applications have become accustomed to relying on these services and implementing various types of environments from complex websites to applications and services provided as SaaS delivery models. Such services and applications are collectively referred to as "cloud applications **115**".

Cloud applications **115** are typically accessed by users using a client device via a web browser. Cloud applications **115** include, among others, e-commerce applications, social media applications, enterprise applications, gaming applications, media sharing applications, storage applications, software development applications, and so on. Many individual users, businesses, and enterprises turn to cloud applications in lieu of "traditional" software applications that are locally installed and managed. For example, an enterprise can use Office® 365 online services for email accounts, rather than having an Exchange® Server maintained by the enterprise.

The networked system **100** further includes a managed network proxy **120**, client devices **130-1** through **130**-N, and a suffix proxy **140** that are communicatively connected to a network **150**. The network **150** may be, for example, a wide area network (WAN), a local area network (LAN), the Internet, and the like. Each of the client devices **130** may include, for example, a personal computer, a laptop, a tablet computer, a smartphone, a wearable computing device, or any other computing device.

The client devices **130** are configured to access the one or more cloud applications **115** executed in the cloud computing platform **110**. A client device **130** may be a managed device or unmanaged device. A managed device is typically secured by an IT personnel of an organization, while an unmanaged device is not. Referring to the above example, the work computer is a managed device while the home computer is an unmanaged device.

The managed network proxy **120** is configured to secure any or all traffic and activities in a cloud computing platform **110**. Specifically, the managed network proxy **120** can be used to intercept, monitor, modify, and forward network communications traffic between client devices **130** and the cloud computing platform **110**.

The managed network proxy **120** can be configured to detect and mitigate network threats against the cloud applications **115** and/or the infrastructure of the cloud computing platform **110**. As non-limiting examples, the managed network proxy **120** can be configured to notify of suspicious network traffic and behavior; block threats; perform application control, URL filtering, and malware protection on the network traffic; establish visibility to application layer parameters (e.g., list of users, devices, locations, etc.); generate profiles of users using the cloud applications **115**; provide alerts on specific or predefined events; generate audit logs; and so on. The architecture and operation of the managed network proxy **120** is discussed in U.S. patent application Ser. No. 14/539,980 assigned to the common assignee, and incorporated herein by reference.

According to certain embodiments, the suffix proxy **140** is configured to keep URLs and web accesses of a proxied webpage within the hold of the managed network proxy **120**. That is, the modifications performed by the suffix proxy **140** for a request to access a webpage of the cloud application **115** allow directing subsequent traffic to the managed network proxy **120**.

In an embodiment, the suffix proxy **140** can be configured to inspect the network traffic and detect cloud-based application's **115** addresses. Examples for such addresses include, for example, uniform resource locators (URLs), uniform resource identifiers (URIs), and so on. As non-limiting examples, the suffix proxy **140** can decompile, deconstruct, or disassemble network traffic for inspection.

In an embodiment, the suffix proxy **140** can be configured to modify webpages and codes (e.g., JavaScript) executed therein and on the cloud-computing platform **110**, so that no network addresses are provided to the client device **130** that would direct the client device **130** to access the cloud application **115** directly. If such a network address is detected, the suffix proxy **140** is configured to rewrite that address, for example, appending a predefined domain name to the original network address. The added domain name may refer or redirect the browser to the managed network proxy **120**. For example, the URL (network address) http://www.somesite.com would be accessed through http://www.somesite.com.network-proxy-service.com. Various embodiments for rewriting network address are disclosed below.

The suffix proxy **140** can be configured to modify any content, including webpages, sent from the cloud application **115**. The suffix proxy **140** can be configured to inspect and/or decompile any content to identify any referred pages and/or URLs present in the content and rewrite those URLs. As non-limiting examples, file types processed can include HTML or JavaScript and responses can include zipped responses or chunked responses.

In one embodiment, for static webpages, for URLs embedded in such webpages a predefined suffix domain name is added. To this end, the suffix proxy **140** is configured to parse HTML webpages and replace the URLs detected using the regular expressions. A static webpage is a webpage that does not contain client-executable script (e.g., JavaScript) code.

According to another embodiment, in order to suffix network addresses in a dynamic webpage, the suffix proxy **140** is configured to analyze and modify code or scripts being loaded to a browser of the client device **130**. For example, JavaScript can be modified by the suffix proxy **140** to wrap any potential generation of network addresses that would directly access the cloud application **115**. If direct access addresses are identified, the script and/or content generated by the script can be modified to rewrite the address to refer to the managed network proxy **120**.

In an embodiment, the suffix proxy **140** is configured to provide a security sandbox which is a runtime component executed over the client device **130**. Certain function of the security sandbox can be performed in the suffix proxy. The security sandbox is labeled as a security sandbox **145**. In an embodiment, the security sandbox **145** is configured to prevent access to the document object model (DOM) of a webpage. In particular, the security sandbox **145** prevents any access and modification to the DOM during run-time of the script. It should be noted that the operation of the security sandbox **145** to prevent access to the DOM does not require any installation of any software, such as plugins, add-ons, and the like in the client device **130** and/or the browser.

Typically, a browser on a client device **130** can execute a script (e.g., JavaScript) that would change the DOM of a webpage during run-time. As a result, it is possible for the client device's **130** browser to create or modify DOM elements with un-suffixed URLs. In order to prevent such action, the security sandbox **145**, and hence the suffix proxy **140**, are configured to restrict the access of any embedded or loaded script code to the DOM.

In an embodiment, the nature of the restriction can be such that changes to URLs in the DOM, by an original script executed in the webpage, are monitored by the security sandbox **145**. The script code monitoring by the suffix proxy **140** can be invoked for read and write accesses to DOM elements. That is, writes of a URL into the DOM are suffixed with the predefined domain name, and reads of a URL from the DOM are un-suffixed. As a result, there can be separation between the URLs seen by "user" code (e.g., the web-applications code) and the browser itself (the DOM, and the JavaScript representation of it). As a result, the original script code can be effectively maintained and controlled by the security sandbox **145** and any communication with the original server (around the proxy) is prevented. It should be noted that an original script is any script embedded in the webpage not dynamically loaded to the webpage.

In certain configurations, a script can be loaded to a webpage after the webpage is rendered on the browser. Such a script is downloaded from a server (originally configured to serve the page) using any of several forms, including inline scripts inside HTML pages and any code, script, or content files. Examples for such files include, for example, JavaScript, Cascading Style Sheets (CSS), and the like.

Typically, the browser of a client device **130** first loads the main HTML page, and then subsequently loads all referenced and inline scripts. Additional, scripts can also be loaded dynamically by the web application, using, for example, the 'eval' statement.

Because dynamic code loading is initially performed by the statically loaded code (or, once loaded, other dynamic code), the security sandbox **145** can take control of execution by modifying the static script code when the webpage is downloaded to the browser. The modifications to the code can be performed in such way that future dynamically loaded code will be modified during run-time and specific changes to the DOM can be intercepted in order to enforce suffixing of certain URLs. This allows the webpage to remain under the control of the suffix proxy **140**.

In an embodiment, the suffix proxy **140** and the security sandbox **145** are configured to modify the dynamic loaded code. The loaded code is received at the suffix proxy **140** which is configured to analyze the code to determine all elements that potentially (explicitly or implicitly) contain, point, or otherwise refer to network addresses (URLs), and replace and/or wrap elements within code that enforces suffixing of the network addresses. The new script code is loaded at the client device's **130** browser. In some embodiments, caching of script codes can be employed to improve performance. The sandbox **145** during run-time resolves the wrappers in order to enforce suffix and un-suffix of network addresses. As noted above, enforcing suffix or network addresses includes suffixing writes of an address (e.g., a URL) into the DOM with a predefined domain name, and un-suffixing any reads of an address from the DOM.

As non-limiting examples, at least the following DOM elements and properties can be wrapped during the creation of the new script code:

Properties of HTML elements that contain URLs, such as "IFRAME", "STYLE", "LINK", "IMG", "AUDIO", "A", "FORM", "BASE" and "SCRIPT", with the properties: "src", "href" and "action". The getAttribute and setAttribute methods of these elements can also be used to set the aforementioned properties.

Properties of HTML elements that can contain a DOM sub-tree (i.e., more HTML). For such elements, the "appendChild" method can be used to add elements (and code) dynamically and the "innerHTML" property can be used to add extra code.

Properties of the "document" object may contain URLs or Hostnames, such as "cookie" and "domain" (both can contain the origin domain of the window). The "write" method can be used to add elements and code to the page.

An "open" method of XMLHttpRequest objects contains a request URL. An "origin" property of "MessageEvent" objects contains the origin hostname. Methods and properties of the "Window" object contain "location", "postMessage", "eval", and "execScript". The "location" redirects the frame to another URL or determines the current location of the frame. The "postMessage" method has an origin argument. The "eval" and "execScript" properties are used to load code dynamically. Other such elements and properties exist, and any or all of them can be wrapped.

In an embodiment, the wrapping of a DOM element, and thus the creation of a new code, is performed using static hooking of the code. In a non-limiting implantation, the static hooking includes: processing and extracting inline scripts in the HTML code of a webpage. Then, any script code is converted to a syntax tree, such as an Abstract Syntax Tree (AST). In an exemplary embodiment, the AST can be generated using the Mozilla® parser. The syntax tree is recursively traversed and calls to wrappers are inserted in certain nodes of the tree to allow for hooking. Finally, the new code is created from the modified nodes (with the inserted class) and sent to the client device's **130** browser. In an embodiment, the new created code can be cached for further usage.

It should be noted that the inserted wrappers can allow for DOM changes to be intercepted during run-time. The wrappers can be applied to cover any or all potential DOM accesses. As non-limiting examples, the wrappers can be applied (inserted) to some or all the following syntax tree (AST) nodes: 'MemberExpression', 'Identifier', 'AssignmentExpression', and 'CallExpression'. For MemberExpression nodes any potential accesses to object properties of DOM objects, subscription operations with non-literal keys, and access to specific properties (for example, obj.src) having a property name matches a white-list of "interesting" properties, are wrapped. In an embodiment, wrappers are inserted to wrap any appropriate object. Thus, some wrappers may not be required. The security sandbox **145** determines if a wrapper should be handled. In most cases, for example, "false positives", the wrapper will decide to do nothing.

For "Identifier" nodes, any potential accesses to a white-list of global Identifiers (which are properties of the window DOM object, e.g., "location") are wrapped. It should be noted that Identifier AST nodes can appear in many unrelated logical positions in the tree. Instances where the Identifier represents access to a global variable are wrapped. This is determined during the traversal step by checking the parent nodes and eliminating all other cases.

For AssignmentExpression node, the "=" and "+=" assignment operators are wrapped, as relevant DOM properties may be strings (URLs). Assignments to previously "marked" MemberExpressions and Identifiers are handled by another wrapper that specifically handles "set" access.

For CallExpression node, CallExpressions where a previously "marked" MemberExpression or Identifier is the callee are handled by another wrapper that specifically handles function calls. A special case exists with the call to "eval", which behaves like a statement, but is represented as a function call in the AST.

According to various embodiments, different wrapper functions can be defined according to the traversal of the syntax tree. The different wrapper functions behave differently during run-time. The wrapper functions include wrapped_get, wrapped_set, and wrapped_call which are used to wrap access to MemberExpressions. The functions wrapped_name_get, wrapped_name_set, and wrapped_name_call are used to access global Identifiers. The function wrapped_eval_param specifically handles the code passed as the parameter of an "eval" call (which can affect the local scope, and thus cannot be decorated).

In some embodiments, the security sandbox **145** is configured to first detect if the wrapper was invoked on relevant objects or properties. Specifically, for "MemberExpression" wrappers, the property name is checked against a white-list, as well as the subscripted object. For "Identifier" wrappers, a white-list is consulted as well. Objects are determined to be of a certain type ("Document", "Window", HTML elements, and so on), and are also compared to global instances when applicable. These comparisons and lookups can be performed efficiently without significant impact on performance in many cases.

In an embodiment, a wrapper call can be processed using any one of various procedures, including, as non-limiting examples: process dynamically loaded code, where the new code (JavaScript code) is sent to a special REST API endpoint of the proxy for translation and caching, as described below. This can occur in wrappers of "appendChild", "innerHTML", "eval", "execScript", and "write". The wrapper can be processed using suffixing or un-suffixing of a URL or hostname. Finally, a false positive wrapper invocation and resume normal execution can also be detected.

In another embodiment, wrapper function handlers that are responsible for handling DOM access to URL related properties or methods can be divided into logical groups. These groups include, for example, 'getters', 'setters', and 'detectors'. The 'getters' handle "get" wrappers. These will un-suffix handled URLs. If a method (JavaScript type "function") is accessed, a "decorator" is returned (see below). The 'setters' handle 'set' wrappers by suffixing assigned URLs. The 'decorators' are handle 'call wrappers that return matching decorator functions for the wrapped methods, which will suffix or un-suffix URLs according to what the decorated method is.

This decorator can be bound to the correct object using the JavaScript "bind" method. In case of "Identifier" wrappers, the correct object is the global object (in some cases, a window). In case of "MemberExpression" wrappers, this is the object being subscripted.

Following is a non-limiting example for code before and after wrapping:

Before

```
var new__src = location + '/image';
some__img.src = new__src;
var w = window.open('test');
eval('test( )');
var loc = x.src;
postMessage('some__message', 'origin');
```

After

```
var new__src = ___WRAPPED__name__get('location', location) +
'/image';
___WRAPPED__set(some__img, 'src', '=', new__src);
var w = ___WRAPPED__call(window, 'open')('test');
eval(___WRAPPED__eval__param(eval, 'test( )'));
var loc = ___WRAPPED__get(x, 'src');
___WRAPPED__name__call('postMessage',
postMessage)('some__message', 'origin');
```

In order to optimize and accelerate the creation of new code and serving of such code to minimize any delay on the client device **130**, a caching mechanism is provided according to the disclosed embodiments. The caching mechanism is implemented at the suffix proxy **140** and configured to improve the overhead of the translation phase. In an embodiment, all elements of translated code (e.g., inline script, file, or dynamic translation request) are cached per server. The entries are keyed by a cryptographic hash of the original code. The cache is shared across users of the proxy. This way, only the first user per-server will experience the impact of the translation phase for commonly loaded scripts. In an embodiment, the dynamic script translation REST endpoint can also be configured to accept a client-side calculated hash and perform a lookup in the cache using it. This can reduce usage of upload bandwidth for the users of the proxy. In this fashion, dynamically generated scripts will almost never be actually sent to the proxy (except for the first time).

The disclosed caching mechanism further caches responses for dynamic translation requests (per hash) returned with "Cache-Control" and "Expires" HTTP headers such that the result will be cached by the client device's **130** browser. In this manner, the same client device **130** will not frequently query the suffix proxy **140** for the same dynamically generated scripts.

According to some embodiments, further optimization of the run-time performance is achieved by creating an optimized fast-path for irrelevant/false-positive wrapper invocations (at the code level). This is achieved by writing the basic wrapper functions with a limited subset of JavaScript to allow optimization by the client device **130** browser (e.g., using a JIT compiler of a browser). In another embodiment, a manual maintenance interface that allows for profiling and detection of the code paths having the most cache hits is provided. This enables removing certain wrappers from the translated code in the suffix proxy **140** cache.

According to various embodiments, the suffix proxy **140** using the security sandbox **145** is configured to implement additional security measures to protect the cloud application **115** and client devices **130**. According to one embodiment, the security measures include configuring the security sandbox **145** to block a third-party content from being added to the DOM and thereby rendered on the client device **130**. The third-party content may include, toolbars, advertisements, malware, and so on. According to this embodiment, any script code downloaded by to a client device **130** is intercepted by the sandbox **145** at the suffix proxy **140**. Then, the code is analyzed and third-party content called by the script code (and subsequently added to the DOM) is removed.

In an embodiment, the suffix proxy **140** using the security sandbox **145** is configured to prevent access to URLs designated in a predefined blacklist, thereby blocking third-party content. The analysis of the code or any attempt to access third party content is performed in run-time as the code is loaded to the client device **130**. The embodiments for analyzing the code and generating a new code respective are discussed in detail above.

In another embodiment, the suffix proxy **140** using the security sandbox **145** is configured to provide a DOM firewall that prevents websites from accessing certain features of the DOM and/or to perform certain operations on the client device **130**. The actions/features that are restricted may include, for example, preventing a website from loading plugins or modifying the setting of the browser, blocking all cross domain accesses between the client device **130** and other domains, and blocking all asynchronous requests between the webpage and the web server. In an embodiment, an alert is generated to the user of the operations that should be taken or are about to be taken. The user may be able to allow or deny any blocking operation.

It should be noted that the analysis of the code or any attempt to access third party content is performed in runtime as the code is loaded to the client device **130**. The embodiments for analyzing the code and generating a new code respective thereto are discussed in detail above.

In yet another embodiment, the security sandbox **145** is configured to encrypt fields included in a webpage rendered on the web browser. These fields may include, for example, text fields, combo boxes, and the like. According this embodiment, an encryption key is generated by the security sandbox **145**. The key is known only to the client device **130** and the security sandbox **145**, but not to the cloud application **115**. The encryption key is provided to the client device **130** through the new code injected to the webpage. As noted above, such code is sent to the browser from the security sandbox **145** upon analysis of a static webpage and/or when an inline script is sent to the security sandbox **145**.

Using the encryption key, any fields shown on the webpage can be encrypted. The encryption of the data (contents of the field) is performed at the client device **130** while the decryption is performed by the security sandbox **145**. In addition, any encrypted data is intercepted by the security sandbox **145**. Then, all text insertions into the DOM are detected and text insertions of encrypted data are replaced with decrypted data (prior to insertion to the actual DOM). As a result of this operation, the original code (provided by the cloud-application **115**) executed by the browser can access only encrypted data. If the code tries to read the decrypted data out of the DOM, the security sandbox **145** intercepts this attempt and encrypts the data back again.

It should be noted that because the original code is sandboxed, execution of the code cannot access injected objects and/or wrapper function and read out the encryption key. As a result, even if the cloud computing platform **110** is hacked, hackers can access only see encrypted data. If hackers inject code into the live site in order to steal the data from the client device **130**, the injected code is processed by the sandbox **145**, and thus cannot access any data residing in the client device **130**.

It should be understood that the embodiments disclosed herein are not limited to the specific architectures illustrated in FIG. **1** and other architectures may be equally used without departing from the scope of the disclosed embodiments. Specifically, the suffix proxy **140** may reside in the cloud computing platform **110**, a different cloud computing platform, or a connectable datacenter. Moreover, in an embodiment, there may be a plurality of suffix proxies **140** operating as described hereinabove and configured to either have one as a standby appliance to take control in a case of failure, or to share the load between them, or to split the functions between them. Furthermore, without departing from the scope of the disclosed embodiments, various functions of the suffix proxy **140** may be implemented by the managed network proxy **120**.

FIG. **2** shows an exemplary and non-limiting flowchart **200** illustrating a method for suffixing network addresses according to one embodiment.

At S**210**, a webpage sent to a client device from a cloud-application is received. The webpage may be sent from an access proxy or any device in the line of traffic between the client device and the cloud application. In an embodiment, the webpage is intercepted by a suffix proxy.

At S**220**, a static network address designated in the received webpage is suffixed. That is, a predefined domain name suffix is added to the network address. The network address or addresses to be suffixed are determined based on a predefined list of URLs.

At S**225**, a piece of code (JavaScript code) is injected into the webpage to later process scripts, code, or content files that are dynamically loaded to the webpage. Examples for such files include, for example, JavaScript, CSS, and the like. The modified webpage is relayed to the client device's browser.

The rendering of the webpage on the client's browser may cause dynamic loading of content from a server (e.g., server running the cloud-application) to the webpage. According to various embodiments, any attempt to load such code (for example, through a static inline script) is detected in order to allow, e.g., the security sandbox **145** to control the execution of the dynamic code, such as static script code, when the webpage is downloaded to the browser.

At S**230**, any code that is dynamically loaded to the webpage is received. For example, such code can be sent from the browser to the suffix proxy **140**. In an embodiment, the received code is cached for future usage. At S**240**, the received code is modified. In an embodiment, S**240** is performed by the suffix proxy. Specifically, the code modification is performed in such way that future dynamically loaded code will be modified during run-time and specific changes to the DOM can be intercepted in order to enforce suffixing of certain network address.

In some embodiments, the enforcing suffixing of network addresses includes suffixing writes of an address (URL) into the DOM with a predefined domain name and un-suffixing any reads of an address (URL) from the DOM.

As discussed in detail above, the code modification includes wrapping certain DOM elements. In a further embodiment, the modification of the code is performed using static hooking techniques discussed in detail above. Step S**240** will result in a new code (an example for such new code is provided above). In an embodiment, the new code is cached for future usage. The various embodiments of the caching mechanism are described above.

At S**250**, the new code is sent to the client device for execution thereon. It should be noted that S**230** and S**240** are performed for any dynamic code, script, or file included in the webpage. It should be emphasized that steps S**230** and S**240** are performed completely during run-time.

FIG. **3** shows an exemplary and non-limiting flowchart **300** illustrating a method for controlling changes to the DOM according to one embodiment.

At S**310**, a webpage sent to a client device from a cloud-application is received. The webpage may be sent from an access proxy or any device in the line of traffic between the client device and the cloud application. In an embodiment, the webpage is intercepted by the suffix proxy.

At S**320**, a piece of code (e.g., JavaScript code) is injected into the webpage to later process scripts, code, or content files that are dynamically loaded to the webpage. In an embodiment, the piece of code maintains an encryption key in the DOM of the webpage. The encryption key is known to the suffix proxy, but not to the cloud-application and/or a provider of the cloud platform.

At S**330**, the modified webpage is sent to the client device. The injected piece of code together with the encryption key allow the user of the client device to encrypt any text field in the webpage.

At S**340**, encrypted text fields inserted into the DOM are intercepted. The recognition of the encrypted text is performed, for example, by searching for a known encryption pattern. At S**350**, any identified encrypted text field is decrypted. Then, at S**360**, the decrypted data of the identified encrypted text fields is inserted into the DOM. It should be noted that any code (e.g., JavaScript code) provided by the cloud application cannot read decrypted data from the DOM. It should be further noted that any attempt to read such data outside of the DOM is intercepted, for example, by the security sandbox **145**. Therefore, the disclosed method for controlling changes to the DOM provides another layer of security to the cloud-application.

FIG. **4** shows an exemplary and non-limiting block diagram of the suffix proxy **140** constructed according to one embodiment. The suffix proxy **140** may be deployed in cloud-computing platforms, data centers, or as a stand-alone network device. The suffix proxy **140** is configured to at least control and enforce access to cloud applications based on access policies described in greater detail above.

The suffix proxy **140** includes a processing system **410** coupled to a memory **415**, and a security sandbox module **420**. The processing system **410** uses instructions stored in the memory **415** to control the operation of the suffix proxy **140**.

The processing system **410** may comprise or be a component of a larger processing system implemented with one or more processors. The one or more processors may be implemented with any combination of general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), field programmable gate array (FPGAs), programmable logic devices (PLDs), controllers, state machines, gated logic, discrete hardware components, dedicated hardware finite state machines, or any other suitable entities that can perform calculations or other manipulations of information.

The processing system **410** may also include machine-readable media for storing software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the one or more processors, cause the processing system **410** to perform the various functions described herein.

The security sandbox module **420** is configured to monitor any changes to the DOM, prevent access to the DOM, and suffix and un-suffix network addresses. As discussed in detail above, the operation of the security sandbox module **420** is performed in run-time, i.e., when the webpage is rendered on the web browser of a client device **130**.

The webpage is provided by the cloud-application **115**. The operation of the security sandbox module **420** is discussed in detail above. In an embodiment, the security sandbox module **420** can be realized as a processing unit having the various structural configurations discussed in detail above.

The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or non-transitory computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPUs"), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

It should be understood that any reference to an element herein using a designation such as "first," "second," and so forth does not generally limit the quantity or order of those elements. Rather, these designations are generally used herein as a convenient method of distinguishing between two or more elements or instances of an element. Thus, a reference to first and second elements does not mean that only two elements may be employed there or that the first element must precede the second element in some manner. Also, unless stated otherwise a set of elements comprises one or more elements. In addition, terminology of the form "at least one of A, B, or C" or "one or more of A, B, or C" or "at least one of the group consisting of A, B, and C" or "at least one of A, B, and C" used in the description or the claims means "A or B or C or any combination of these elements." For example, this terminology may include A, or B, or C, or A and B, or A and C, or A and B and C, or **2**A, or **2**B, or **2**C, and so on.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiments and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

What is claimed is:

1. A method for modifying network addresses of at least one cloud application, comprising:

receiving a webpage sent to a client device from the at least one cloud application, wherein a the received webpage designates at least one script loaded to the client device during runtime;

injecting, by a suffix proxy device, a piece of code to the received webpage, wherein the suffix proxy device is connected between the client device and a cloud platform hosting the at least one cloud application;

receiving, by the injected piece of code, an indication on an attempt to load each of the at least one script;

modifying the at least one script by suffixing each network address designated in the at least one script with a predefined network address, wherein the suffixing includes wrapping at least one instruction in the at least one script to call an alternative instruction configured for the suffixing; and

sending the modified at least one script to the client device, wherein runtime execution of the modified at least one script on the client device causes redirection of future requests from the client device to the cloud application to the suffixed network address.

2. The method of claim **1**, further comprising:

modifying each static network address designated in the received webpage.

3. The method of claim **2**, wherein suffixing each static network address further comprises:

parsing the webpage to identify the static network address; and

suffixing each identified static network address designated in the at least one script with the predefined network address.

4. The method of claim **3**, wherein each of the network addresses to be modified is included in a list of network addresses.

5. The method of claim **4**, wherein each of the network addresses to be modified is at least a uniform resource locator (URL).

6. The method of claim **1**, wherein modifying the at least one script further comprises: parsing the at least one script to identify at least one instruction configured for generating a network address.

7. The method of claim **6**, further comprising:

detecting, during runtime, at least one wrapped instruction that does not affect a network address; and

ignoring the respective wrapper applied on the at least one wrapped instruction that does not affect the network address.

8. The method of claim **6**, wherein the at least one script is any one of: a JavaScript code and a Cascading Style Sheets (CSS) file.

9. The method of claim **6**, wherein wrapping each of the at least one identified instruction further comprises:

converting the at least one script into an abstract syntax tree;

recursively traversing the abstract syntax tree; and

wrapping nodes of the abstract syntax tree to generate a patched abstract syntax tree.

10. The method of claim **9**, further comprising:

rebuilding the modified at least one script from the patched abstract syntax tree.

11. The method of claim **6**, further comprising:

caching the at least one script and the modified script.

12. The method of claim **1**, further comprising:

determining if at least one modified script has been cached upon identifying a request to load the at least one script; and

sending the cached modified script to the client device without retrieving a webpage requested by the client device from a remote server executing the at least one cloud application.

13. The method of claim **1**, wherein the method is performed without requiring any modification of a default client device configuration.

14. The method of claim **6**, further comprising:

executing in a sandbox the at least one script to monitor any attempts to change a document object model (DOM) of the webpage;

suffixing, at runtime, any write of a network address to the DOM; and

un-suffixing, at runtime, any read of a network address from the DOM.

15. The method of claim **14**, further comprising:

monitoring access to the DOM; and

preventing at least one operation that can modify the DOM.

16. The method of claim **15**, wherein the at least one operation includes any one of: loading plugins to the client device, a cross domain access, and an asynchronous request between the webpage and a web server.

17. The method of claim **1**, wherein the method is performed by a suffix proxy device.

18. A non-transitory computer readable medium having stored thereon instructions for causing one or more processing units to execute the method according to claim **1**.

19. A system for modifying network addresses of at east one cloud application, comprising:

a processor; and

a memory containing instructions that, when executed by the processor, configure the system to:

receive a webpage sent to a client device from the at east one cloud application, wherein a the received webpage designates at east one script loaded to the client device during runtime;

inject, by the system, a piece of code to the received webpage, wherein the suffix proxy device is connected between the client device and a cloud platform hosting the at least one cloud application;

receive, by the injected piece of code, an indication on an attempt to load each of the at least one script;

modify the at least one script by suffixing each network address designated in the at least one script with a predefined network address, wherein the suffixing includes wrapping at least one instruction in the at least one script to call an alternative instruction configured for the suffixing and

send the modified at least one script to the client device, wherein runtime execution of the modified at least one script on the client device causes redirection of future requests from the client device to the cloud application to the suffixed network address.

20. The system of claim **19**, wherein the system is further configured to:

modify each static network address designated in the received webpage.

21. The system of claim **20**, wherein the system is further configured to:

parse the webpage to identify the static network address; and

suffix each identified static network address designated in the at least one script with the predefined network address.

22. The system of claim **21**, wherein each of the network addresses to be modified is included in a list of network addresses.

23. The system of claim **22**, wherein each of the network addresses to be modified is at least a uniform resource locator (URL).

24. The system of claim **19**, wherein the system is further configured to: parse the at least one script to identify at least one instruction configured for generating a network address.

25. The system of claim **24**, wherein the system is further configured to:

detect, during run-time, at least one wrapped instruction that does not affect a network address; and

ignore the respective wrapper applied on the at least one wrapped instruction that does not affect the network address.

26. The system of claim **24**, wherein the at least one script is any one of: a JavaScript code and a Cascading Style Sheets (CSS) file.

27. The system of claim **24**, wherein the system is further configured to:

convert the at least one script into an abstract syntax tree;

recursively traverse the abstract syntax tree; and

wrap nodes of the abstract syntax tree to generate a patched abstract syntax tree.

28. The system of claim **27**, wherein the system is further configured to:

rebuild the modified at least one script from the patched abstract syntax tree.

29. The system of claim **24**, wherein the system is further configured to:

cache the at least one script and the modified script.

30. The system of claim **19**, wherein the system is further configured to:

determine if at least one modified script has been cached upon identifying a request to load the at least one script; and

send the cached modified script to the client device without retrieving a webpage requested by a client device from a remote server executing the at least one cloud application.

31. The system of claim **24**, wherein the system is further configured to:

execute in a sandbox the at least one script to monitor any attempts to change a document object model (DOM) of the webpage;

suffix, at run-time, any write of a network address to the DOM; and

un-suffix, at run-time, any read of a network address from the DOM.

32. The system of claim **31**, wherein the system is further configured to:

monitor access to the DOM; and

prevent at least one operation that can modify the DOM.

33. The system of claim **32**, wherein the at least one operation includes any one of: loading plugins to the client device, a cross domain access, and an asynchronous request between the webpage and a web server.

* * * * *

EXHIBIT B

This copy is for your personal, non-commercial use only. Distribution and use of this material are governed by our Subscriber Agreement and by copyright law. For non-personal use or to order multiple copies, please contact Dow Jones Reprints at 1-800-843-0008 or visit www.djreprints.com.

https://www.wsj.com/articles/microsoft-patched-bing-vulnerability-that-allowed-snooping-on-email-and-other-data-25b58831

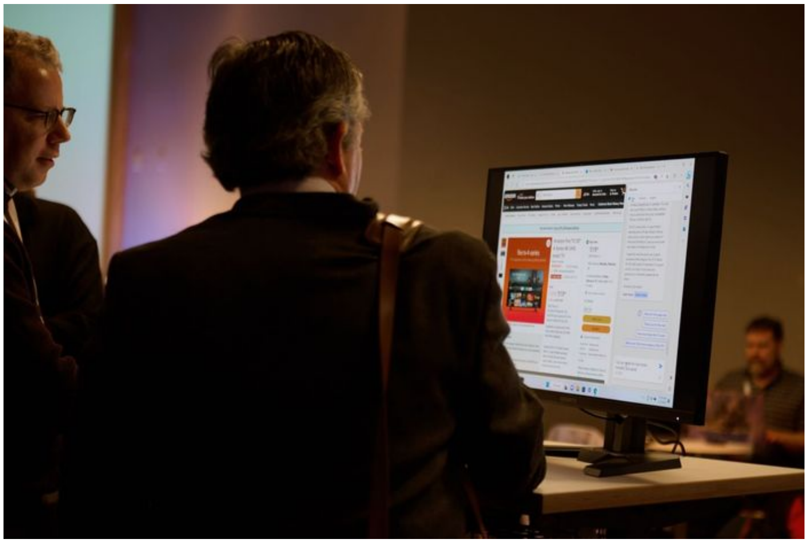
◆ WSJ NEWS EXCLUSIVE | TECH

Microsoft Patched Bing Vulnerability That Allowed Snooping on Email and Other Data

The issue was fixed days before the software company launched Bing with AI

By Robert McMillan Follow

Updated March 29, 2023 4:34 pm ET



Microsoft has been adding generative AI capabilities to much of its software and services. PHOTO: CHONA KASINGER/BLOOMBERG NEWS

Microsoft Corp. MSFT **1.46%** ▲ patched a dangerous security issue in Bing last month, days before it launched a new artificial intelligence-powered version of the search engine.

The problem was discovered by outside researchers at the security firm Wiz Inc. It was created by a mistake in the way that Microsoft configured applications on Azure, its cloud-computing platform, and could be exploited to gain access to emails and other documents of people who used Bing, the researchers said.

Microsoft fixed the problem on Feb. 2, according to Ami Luttwak, Wiz's chief technology officer. Five days later Satya Nadella introduced the new generative-AI capabilities to Bing, bringing a renewed interest in Microsoft's 14-year-old search engine. Usage of Bing has jumped, rising to more than 100 million daily active users in the month since the upgrade, Microsoft said in a recent blog.

Microsoft has been adding generative-AI capabilities to much of its software and services. The new Bing can help users track down information using a chatbot backed by the technology behind ChatGPT.

Microsoft is adding the technology to its popular Microsoft 365 suite of business software. This week it unveiled plans to use AI to help cybersecurity experts monitor and categorize threats and attacks.

A Microsoft spokesman said the misconfiguration issue affected a small number of the company's applications that used its login management service, called Azure Active Directory.

"We appreciate the collaboration with Wiz, which helped us mitigate a potential risk and further harden our services and thank them for working with us to protect the ecosystem," the company said.

In a blog post, Microsoft said the issues pointed out by Wiz had been fixed and outlined ways for companies and consumers to protect themselves. The company said it had taken steps to prevent this type of issue from occurring in the future.

Microsoft shares closed nearly 2% higher Wednesday, in line with the rise of the Nasdaq Composite Index.

Wiz said there is no evidence anyone has taken advantage of the issue. It isn't clear how long it was available for hackers to use, although the issue may have been exploitable for years, the cybersecurity company said.

Hillai Ben-Sasson, a researcher at Wiz, said the misconfiguration allowed him to access a website used by Microsoft employees to set up trivia quizzes on Bing. Because it was misconfigured, anyone with a free Microsoft account could use it to change what results popped up on Bing for search queries.

It should only have been viewable to Microsoft employees, Wiz's Mr. Luttwak said. "We should have never seen it," he said.

The Wiz team discovered they could change some Bing search results by changing data on the Bing trivia page. They were able to make specific results show up for any search query by tinkering with the trivia page. They made the 1995 film "Hackers" pop up for anyone who searched for the term "best soundtracks."

Then they discovered something more serious: a way to get access to Bing users' Microsoft 365 emails, documents, calendars and other data.

This kind of access would be extremely valuable to hackers who could use it to steal sensitive information, send fraudulent emails and gain access to computer systems.

"A potential attacker could have influenced Bing search results and compromised Microsoft 365 emails and data of millions of people," Mr. Luttwak said. "It could have been a nation-state trying to influence public opinion or a financially motivated hacker."

In addition to the trivia site, Wiz researchers found about 1,000 other websites on Microsoft's cloud that appeared to have similar problems. Most of the pages looked like they belonged to Azure customers but at least 10 of them were Microsoft's.

Microsoft has emerged as one of the world's largest cybersecurity companies. It has also been plagued by security issues recently as it tries to lock down both its legacy products, which run on personal computers and in corporate data centers while integrating them with its fast-growing cloud computing platform.

Write to Robert McMillan at robert.mcmillan@wsj.com

Appeared in the March 30, 2023, print edition as 'Microsoft Patched a Bing Security Issue'.

EXHIBIT C

Orca Security vs. Wiz Comparison Chart

Orca Security	Wiz	±
Learn More	Learn More	ADD TO COMPARE
Update Features	Update Features	

Platforms Supported

Windows	✓
Mac	✓
Linux	✓
SaaS / Web	✅
On-Premises	✅
iPhone	✓
iPad	✓
Android	✓
Chromebook	✓

Audience

Enterprise organizations and cloud companies

Support

Business Hours	✅
24/7 Live Support	✅
Online	✅

API

Offers API	✓

Platforms Supported

Windows	✓
Mac	✓
Linux	✓
SaaS / Web	✅
On-Premises	✓
iPhone	✓
iPad	✓
Android	✓
Chromebook	✓

Audience

Enterprises interested in a cloud infrastructure security solution to find critical risks and infiltration vectors

Support

Business Hours	✅
24/7 Live Support	✅
Online	✅

API

Offers API	✅

Related Products

Heimdal Endpoint Detection and Response (EDR)

Heimdal® Endpoint Detection and Response is our proprietary multi-solution service providing unique prevention, threat-hunt...

★★★★½ 54 Ratings

Visit Website

astra pentest

Astra Pentest

Astra's Pentest is a comprehensive penetration testing solution with an intelligent automated vulnerability scanner coupled...

★★★★½ 45 Ratings

Visit Website

Safetica

Safetica is a cost-effective, easy-to-use Data Loss Prevention (DLP) solution. It performs security audits, prevents sensitive...

★★★★½

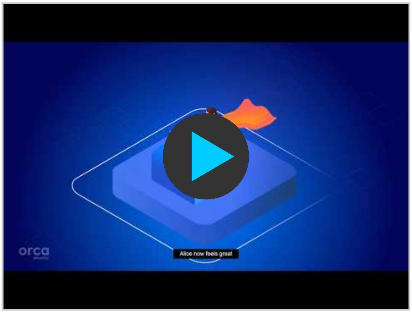
263 Ratings

Visit Website

satori

Satori

Screenshots and Videos



View more images or videos

Pricing

No information available.

Free Version	✓
Free Trial	✓

Reviews/Ratings

Overall ★★★★★

Ease ★★★★★

Features ★★★★★

Design ★★★★★

Support ★★★★★

This software hasn't been reviewed yet. Be the first to provide a review:

Review this Software

Training

Documentation	✓
Webinars	✓
Live Online	✓
In Person	✓

Screenshots and Videos

Created and Resolved Issues
High and Critical Severity Issues
Open Issu
Issues
Patch Management Issues Trend
Top Patch Management Issues

View more images or videos

Pricing

No information available.

Free Version	✓
Free Trial	✓

Reviews/Ratings

Overall ★★★★★

Ease ★★★★★

Features ★★★★★

Design ★★★★★

Support ★★★★★

This software hasn't been reviewed yet. Be the first to provide a review:

Review this Software

Training

Documentation	✓
Webinars	✓
Live Online	✓
In Person	✓

Satori created the first DataSecOps platform which streamlines data access by automating access controls, secu-...

★★★★★ 58 Ratings

Visit Website

Kasm Workspaces

Kasm Workspaces is a container streaming platform for delivering browser, desktop and application workloads to the web browser....

★★★★★

117 Ratings

Visit Website

Kloudle

Kloudle automates cloud security for devs, DevOps & engineering teams. View all cloud assets of AWS, Google Cloud, Azure,...

★★★★★ 31 Ratings

Visit Website

Finite State

Finite State manages risk across the software supply chain with comprehensive SCA and SBOMs for the connected world. By pro...

★★★★★ 1 Rating

Company Information

Orca Security

Founded: 2019

United States

orca.security/

Alternatives

Kloudle
★★★★★

Sonrai Security
Sonraí Security
★★★★☆

Cloudnosys

Wiz

Uptycs

View All

Company Information

Wiz

Founded: 2020

United States

www.wiz.io

Alternatives

Sonrai Security
Sonraí Security
★★★★☆

Lacework

Orca Security

Runecast
Runecast Solutions

Cloudaware

View All

Visit Website

FlashStart

FlashStart is a global, cloud-based cyber security platform that specializes in DNS filtering with the support of artificial in-...

★★★★★ 85 Ratings

Visit Website

Critical Start

Our team of highly trained cyber-security professionals provides expertise in compliance, tool as-sessments, threat hunting, inci-...

★★★★★ 2 Ratings

Visit Website

ConnectWise Cybersecurity Management

Define and Deliver Comprehensive Cybersecurity Services. Security threats con-tinue to grow, and your clients...

★★★★☆

153 Ratings

Visit Website

Categories

- Cloud Detection and Response (CDR) ✓
- Cloud Security ✓
- Cloud Workload Protection ✓
- Cloud-Native Application Protection Platforms (CNAPP) ✓
- Cybersecurity ✓
- Vulnerability Management ✓
- Vulnerability Scanners ✓

Categories

- Cloud Detection and Response (CDR) ✓
- Cloud Security ✓
- Cloud Security Posture Management (CSPM) ✓
- Cloud Workload Protection ✓
- Cloud-Native Application Protection Platforms (CNAPP) ✓
- Container Security ✓
- Cybersecurity ✓
- Data Security Posture Management (DSPM) ✓
- Identity Management ✓
- IT Security ✓
- Patch Management ✓
- Software Asset Management (SAM) ✓
- Vulnerability Management ✓
- Vulnerability Scanners ✓

Cloud Security Features

Antivirus	✅
Application Security	✅
Behavioral Analytics	✅
Encryption	✅
Endpoint Management	✅
Incident Management	✅
Intrusion Detection System	✅
Threat Intelligence	✅
Two-Factor Authentication	✓
Vulnerability Management	✅

Cybersecurity Features

AI / Machine Learning	✓
Behavioral Analytics	✓
Endpoint Management	✅
Incident Management	✅
IOC Verification	✅
Tokenization	✓
Vulnerability Scanning	✅
Whitelisting / Blacklisting	✓

Vulnerability Management Features

Asset Discovery	✅
Asset Tagging	✅
Network Scanning	✅
Patch Management	✅
Policy Management	✅
Prioritization	✅
Risk Management	✅

Cloud Security Features

Antivirus	✅
Application Security	✅
Behavioral Analytics	✅
Encryption	✅
Endpoint Management	✅
Incident Management	✅
Intrusion Detection System	✅
Threat Intelligence	✅
Two-Factor Authentication	✓
Vulnerability Management	✅

Cloud Workload Protection Features

Anomaly Detection	✓
Asset Discovery	✓
Cloud Gap Analysis	✅
Cloud Registry	✓
Data Loss Prevention (DLP)	✓
Data Security	✓
Governance	✅
Logging & Reporting	✓
Machine Learning	✓
Security Audit	✓
Workload Diversity	✓

Cybersecurity Features

AI / Machine Learning	✓
Behavioral Analytics	✓
Endpoint Management	✅
Incident Management	✅

Vulnerability Assessment	✅
Web Scanning	✅

IOC Verification	✅
Tokenization	✓
Vulnerability Scanning	✅
Whitelisting / Blacklisting	✓

Vulnerability Management Features

Asset Discovery	✅
Asset Tagging	✅
Network Scanning	✅
Patch Management	✅
Policy Management	✅
Prioritization	✅
Risk Management	✅
Vulnerability Assessment	✅
Web Scanning	✅

Vulnerability Scanners Features

Asset Discovery	✓
Black Box Scanning	✓
Compliance Monitoring	✅
Continuous Monitoring	✓
Defect Tracking	✓
Interactive Scanning	✓
Logging and Reporting	✓
Network Mapping	✓
Perimeter Scanning	✓
Risk Analysis	✅
Threat Intelligence	✅
Web Inspection	✓

Show More Features

Integrations

Amazon Web Services (AWS)	✓
Enso	✓
Google Cloud Platform	✓
Jira Software	✓
Jira Work Management	✓
Microsoft Azure	✓
Okta	✓
Seemplicity	✓
ServiceNow	✓
Silk Security	✓

Show More Integrations

View All 19 Integrations

Claim Orca Security and update features and information

Integrations

Amazon Web Services (AWS)	✓
Enso	✓
Google Cloud Platform	✓
Jira Software	✓
Jira Work Management	✓
Microsoft Azure	✓
Okta	✓
Seemplicity	✓
ServiceNow	✓
Silk Security	✓

Show More Integrations

View All 22 Integrations

Claim Wiz and update features and information

SourceForge

Open Source Software

Business Software

Add Your Software

Business Software Advertising

Company

About

Team

SourceForge Headquarters
225 Broadway Suite 1600
San Diego, CA 92101
+1 (858) 454-5900

Resources

Support

Site Documentation

Site Status

© 2023 Slashdot Media. All Rights Reserved.

EXHIBIT D

US 20170124622A1

(19) **United States**
(12) **Patent Application Publication**
KLIPER

(10) **Pub. No.: US 2017/0124622 A1**
(43) **Pub. Date: May 4, 2017**

(54) **SYSTEM AND METHOD FOR INTUITIVE CONTENT BROWSING**

(71) Applicant: **The Joan and Irwin Jacobs Technion-Cornell Institute**, New York, NY (US)

(72) Inventor: **Roi KLIPER**, New York, NY (US)

(73) Assignee: **The Joan and Irwin Jacobs Technion-Cornell Institute**, New York, NY (US)

(21) Appl. No.: **15/404,827**

(22) Filed: **Jan. 12, 2017**

Related U.S. Application Data

(63) Continuation-in-part of application No. 14/940,396, filed on Nov. 13, 2015.

(60) Provisional application No. 62/279,125, filed on Jan. 15, 2016, provisional application No. 62/147,771, filed on Apr. 15, 2015, provisional application No. 62/079,804, filed on Nov. 14, 2014.

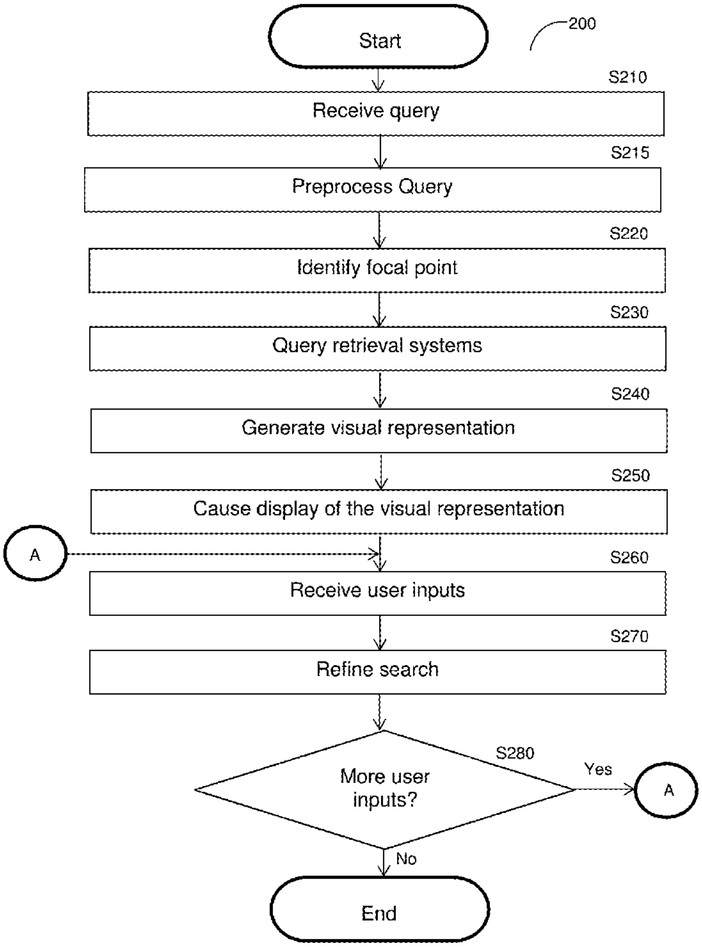
Publication Classification

(51) **Int. Cl.**
G06Q 30/06 (2006.01)
G06F 17/30 (2006.01)

(52) **U.S. Cl.**
CPC ***G06Q 30/0625*** (2013.01); ***G06Q 30/0641*** (2013.01); ***G06F 17/30905*** (2013.01); *G06F 3/0485* (2013.01)

(57) **ABSTRACT**

A method and system for intuitive content browsing. The method includes determining, based on a request to browse content, an initial focal point for a visual representation of the content, wherein the initial focal point represents a content item; identifying, based on the request and the determined initial focal point, the content to be browsed; generating, based on the identified content and the focal point, a visual representation of the identified content, wherein the generated visual representation includes the identified content organized with respect to the initial focal point; and sending, to a user device, the generated visual representation for display, wherein the identified content is browsed via the displayed visual representation with respect to the focal point.



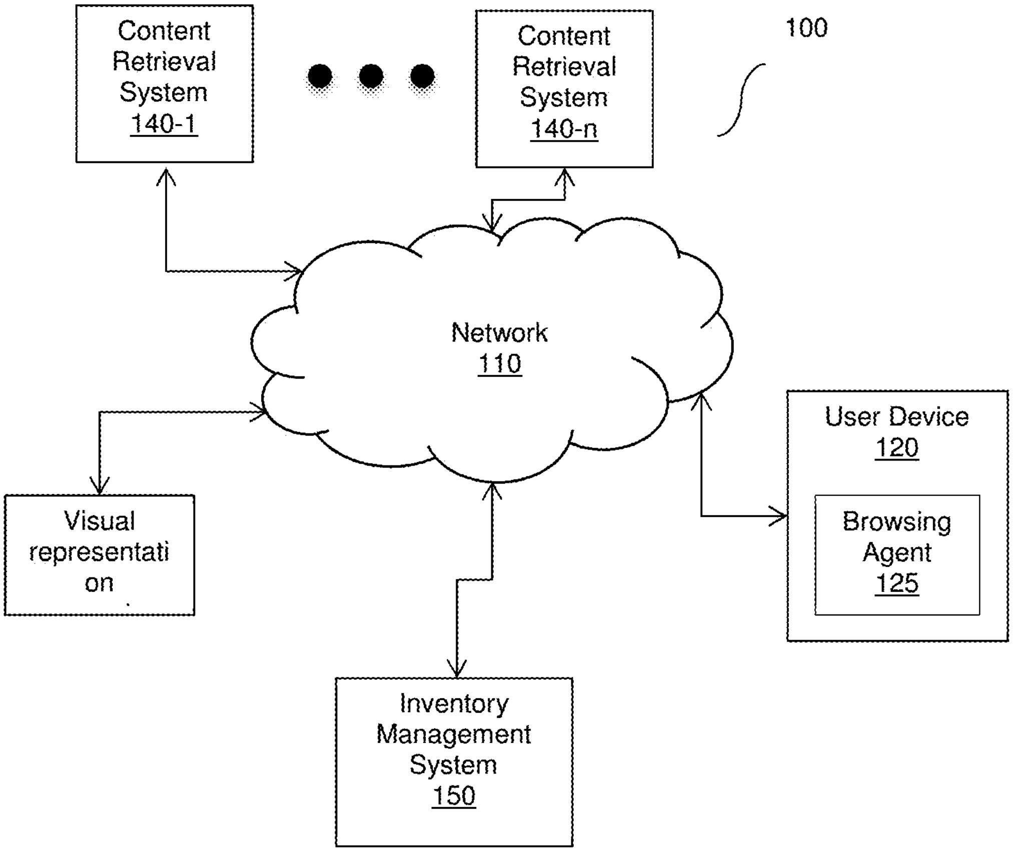

Content Retrieval System 140-1
Content Retrieval System 140-n
100
Network 110
Visual representati on
User Device 120
Browsing Agent 125
Inventory Management System 150

FIG. 1

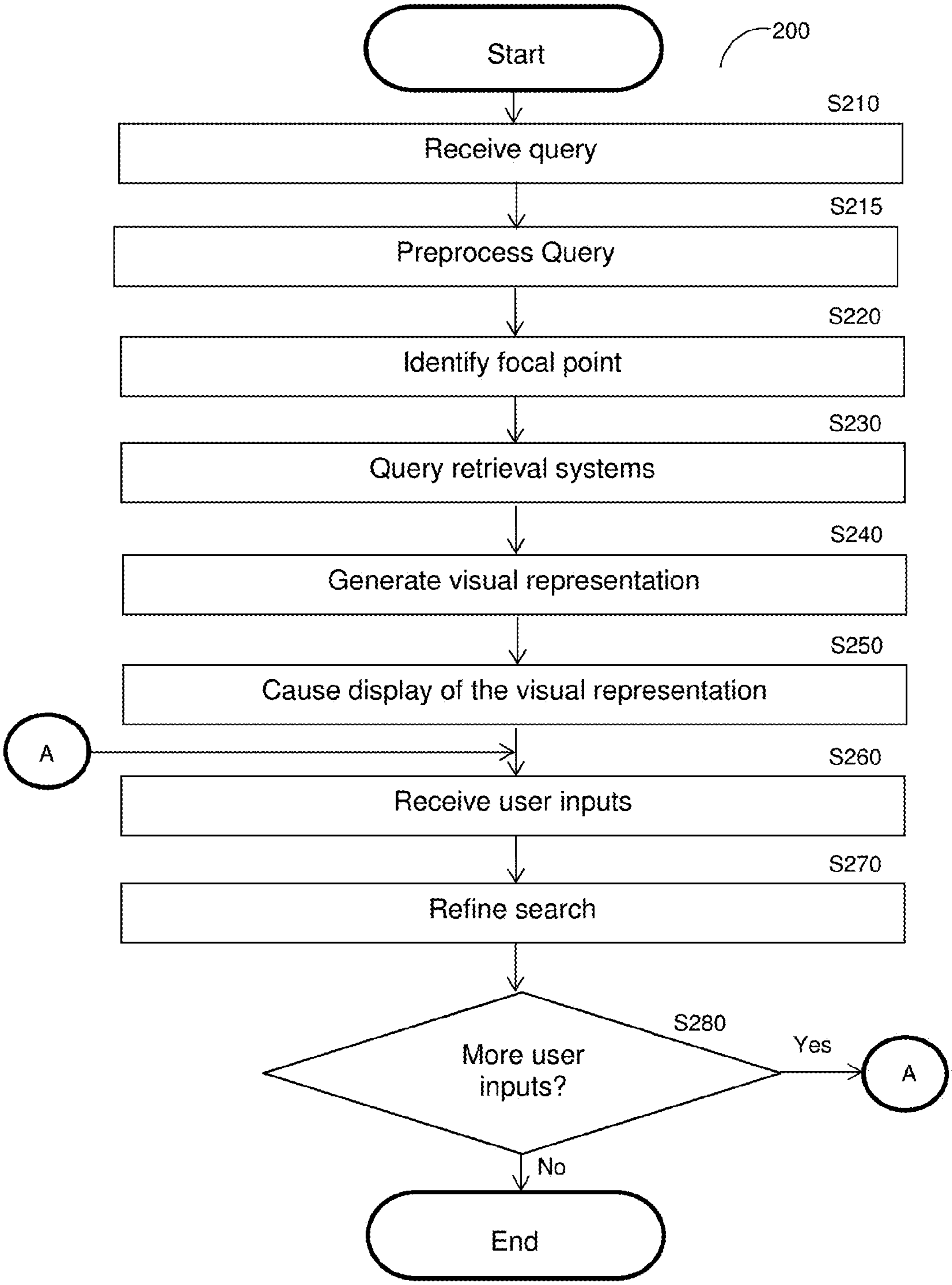

Start
200
S210
Receive query
S215
Preprocess Query
S220
Identify focal point
S230
Query retrieval systems
S240
Generate visual representation
S250
Cause display of the visual representation
A
S260
Receive user inputs
S270
Refine search
S280
More user inputs?
Yes
A
No
End

FIG. 2

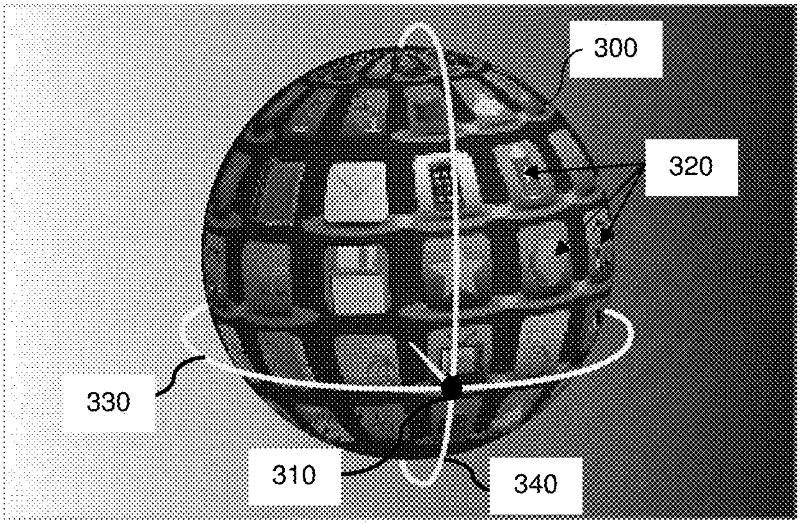


FIG. 3

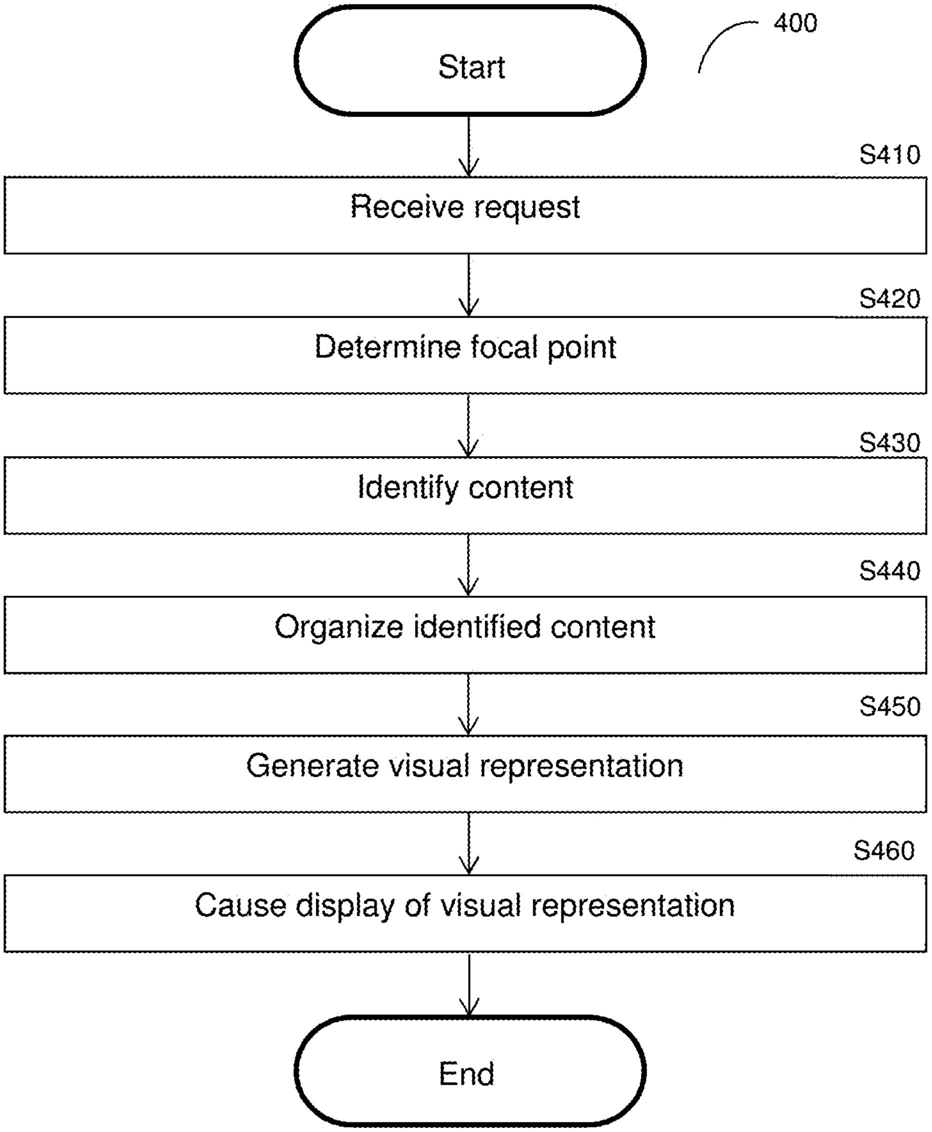

400
Start
S410
Receive request
S420
Determine focal point
S430
Identify content
S440
Organize identified content
S450
Generate visual representation
S460
Cause display of visual representation
End

FIG. 4

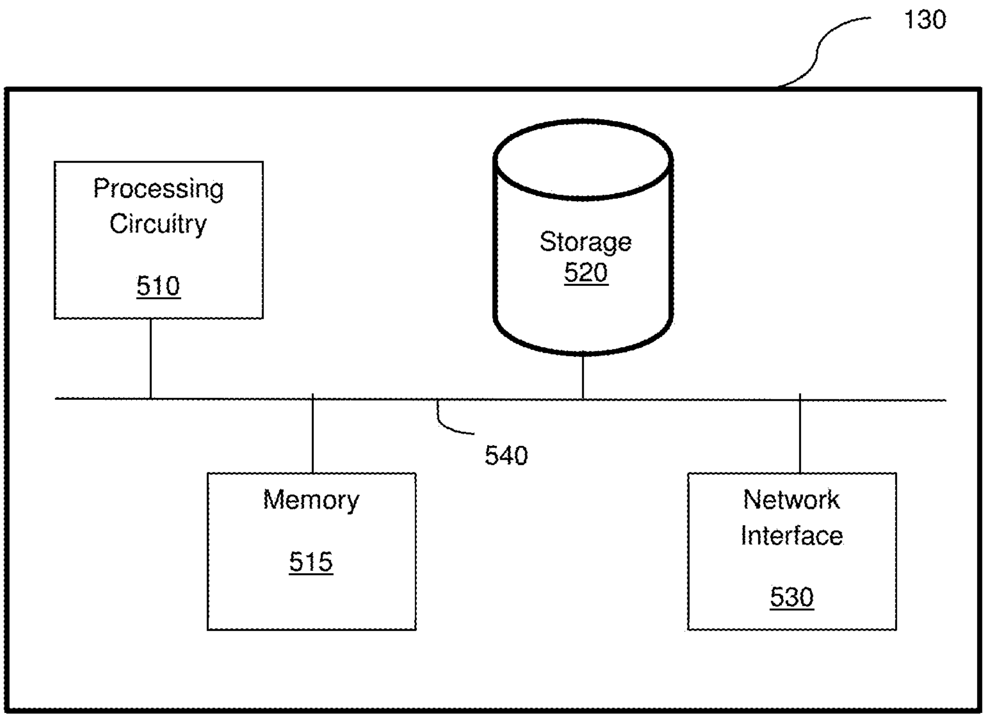

130
Processing Circuitry
510
Storage
520
540
Memory
515
Network Interface
530

FIG. 5

SYSTEM AND METHOD FOR INTUITIVE CONTENT BROWSING

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 62/279,125 filed on Jan. 15, 2016. This application is also a continuation-in-part of U.S. patent application Ser. No. 14/940,396 filed on Nov. 13, 2015, now pending. The Ser. No. 14/940,396 Application claims the benefit of U.S. Provisional Application No. 62/147,771 filed on Apr. 15, 2015, and of U.S. Provisional Application No. 62/079,804 filed on Nov. 14, 2014. The contents of the above-referenced applications are hereby incorporated by reference.

TECHNICAL FIELD

[0002] The present disclosure relates generally to displaying content, and more particularly to intuitively organizing content to allow for interactions in two-dimensional and three-dimensional space.

BACKGROUND

[0003] As the Internet becomes increasingly prevalent in modern society, the amount of information available to the average user has increased astronomically. Consequently, systems for retrieving and browsing web-based content are used much more frequently. Such systems often accept a user query for content to browse in the form of, for example, text, multimedia content (an image, videos, audio), and so on. The widespread availability of new content has led to further developments of display mechanisms allowing users to consumer and interact with data. For example, touch screens allowing users to intuitively interact with displayed content and three-dimensional virtual reality displays allowing users to view content in an immersive environment have become available to the average person. These evolving display mechanisms provide varied and improved user experiences as time goes on.

[0004] To obtain relevant content, user queries must contain sufficient information to identify relevant material. Although algorithms used in tandem with, e.g., search engines, have been developed to provide a much greater likelihood of finding relevant content for even basic queries, users may nevertheless face challenges in accurately finding particularly relevant content due to the arcane rules utilized in accepting user queries. For example, users can find more accurate content using logical operators that may not be known or understood by the average person.

[0005] The challenges in utilizing existing content retrieval systems cause further difficulties for users seeking to refine queries. Refinement may include submitting a refined query to the retrieval system and receiving new results respective of the refined query, thereby effectively submitting a new search. As a result, refinement wastes time and computing resources due to the submission of additional queries, even for users that are familiar with the idiosyncrasies of web-based content searches. Further, inexperienced users may be frustrated by the inability to properly refine their searches to obtain the desired results.

[0006] As an example, a user living in New York City seeking to purchase wine may submit a query of "wine." Upon viewing search results related to wine generally, the user may wish to refine his search to focus on red wine and, as a result, enters a refined query of "red wine." The user may wish to further refine his search to focus on red wine originating from France and, thus, enter a refined query of "red wine France." The results of this search may include content related red wine being sold in France and/or to red wine originating from France being sold anywhere in the world. The user may further need to refine his search on French red wine that can be bought locally and, therefore, enter a further refined query of "red wine France in New York." Each of the refinements requires the user to manually enter a refined query and submit the query for a new search, thereby wasting the user's time and unnecessarily using computing resources.

[0007] Existing solutions for refining content queries often involve offering predetermined potential refined queries and directing users to content upon user interactions with the potential refined queries. The potential refined queries may be based on, e.g., queries submitted by previous users. However, previous user queries do not always accurately capture a user's current needs, particularly when the user is not aware of his or her needs. For example, a user seeking to buy chocolate may initially enter the query "chocolate" before ultimately deciding that she would like to buy dark chocolate made in Zurich, Switzerland. Potential refinements offered based on the initial query may include "dark chocolate," "white chocolate," "milk chocolate," and "Swiss chocolate," none of which entirely captures the user's ultimate needs. Thus, the user may need to perform several refinements and resend queries multiple times before arriving at the desired content.

[0008] Moreover, for users seeking to purchase products, it may be difficult to determine in which stores the products are physically available. To find such information, a user may need to visit e-commerce websites of stores until he or she finds a store that lists the item as "in stock." Nevertheless, such listings may be outdated or otherwise inaccurate, thereby causing user frustration and a need to conduct further searches.

[0009] Further, when viewing search results or otherwise viewing content, the user is typically presented with display options such as organizing content in various organizational schemes (e.g., list form, grid form, and so on) and/or based on different ordering schemes (e.g., by date or time, relevancy to a query, alphabetical order, and so on). For example, a user viewing content related to a particular book may wish to view content related to books by the same author, about the same subject, from the same genre or literary era, and so on. To view this additional content, users may be able to reorganize displayed content by, e.g., changing the organizational scheme, submitting refinement queries, changing the ordering scheme, and so on.

[0010] To this end, it is desirable to provide content in ways that are intuitive and therefore easily digestible by the average user. Intuitive content organization and navigation therefore serve an important role in improving the overall user experience by increasing user engagement and allowing for more efficient retrieval and/or viewing of content. Such improvements to user experience may be particularly important in the search engine context, as improved user experience may result in increased use of search engine services and/or purchases of products.

[0011] Additionally, some solutions exist for allowing users to browse stores remotely via remoted controlled on premises cameras (i.e., disposed in the store) or preexisting (e.g., static) photos or images of the premises and inventory. These solutions allow users to intuitively engage with content as they would in a physical store, but are typically limited to displaying store contents as they were previously (based on previously captured images) or as they currently are (e.g., via live video feed or images otherwise captured in real-time), but not based on potential future configurations (e.g., based on predicted inventories and other future changes).

[0012] It would therefore be advantageous to provide a solution that would overcome the deficiencies of the prior art.

SUMMARY

[0013] A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the term "some embodiments" may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

[0014] Certain embodiments disclosed herein include a method for intuitive content browsing. The method includes determining, based on a request to browse content, an initial focal point for a visual representation of the content, wherein the initial focal point represents a content item; identifying, based on the request and the determined initial focal point, the content to be browsed; generating, based on the identified content and the focal point, a visual representation of the identified content, wherein the generated visual representation includes the identified content organized with respect to the initial focal point; and sending, to a user device, the generated visual representation for display, wherein the identified content is browsed via the displayed visual representation with respect to the focal point.

[0015] Certain embodiments disclosed herein also include a non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to perform a process, the process comprising: determining, based on a request to browse content, an initial focal point for a visual representation of the content, wherein the initial focal point represents a content item; identifying, based on the request and the determined initial focal point, the content to be browsed; generating, based on the identified content and the focal point, a visual representation of the identified content, wherein the generated visual representation includes the identified content organized with respect to the initial focal point; and sending, to a user device, the generated visual representation for display, wherein the identified content is browsed via the displayed visual representation with respect to the focal point.

[0016] Certain embodiments disclosed herein also include a system for intuitive content browsing. The system comprises: a processing circuitry; and a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: determine, based on a request to browse content, an initial focal point for a visual representation of the content, wherein the initial focal point represents a content item; identify, based on the request and the determined initial focal point, the content to be browsed; generate, based on the identified content and the focal point, a visual representation of the identified content, wherein the generated visual representation includes the identified content organized with respect to the initial focal point; and send, to a user device, the generated visual representation for display, wherein the identified content is browsed via the displayed visual representation with respect to the focal point.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

[0018] FIG. **1** is a network diagram utilized to describe the various disclosed embodiments.

[0019] FIG. **2** is a flowchart illustrating a method for organizing content according to an embodiment.

[0020] FIG. **3** is a screenshot illustrating a spherical organization of content.

[0021] FIG. **4** is a flowchart illustrating a method for displaying content that may be intuitively browsed according to an embodiment.

[0022] FIG. **5** is a schematic diagram of a visual representation generator according to an embodiment.

DETAILED DESCRIPTION

[0023] It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

[0024] FIG. **1** shows an example network diagram **100** utilized to describe the various disclosed embodiments. The network diagram **100** includes a network **110**, a user device **120**, a visual representation generator **130**, a plurality of content retrieval systems **140-1** through **140-*n*** (hereinafter referred to individually as a search engine **140** and collectively as search engines **140**, merely for simplicity purposes) and an inventory management system **150**.

[0025] The network **110** may be the Internet, the world-wide-web (WWW), a local area network (LAN), a wide area network (WAN), a metro area network (MAN), and other networks configured to communicate between the elements of the **110**. The user device **120** may be a personal computer (PC), a personal digital assistant (PDA), a mobile phone, a smart phone, a tablet computer, a wearable computer device, an e-reader, a game console, or any other device equipped with browsing capabilities. The content retrieval systems **140** may include, but are not limited to, search engines or other sources of content from which content may be retrieved. Alternatively or collectively, the content retrieval systems **140** may include or be communicatively connected to one or more data sources which can be queried or crawled for content.

[0026] The user device **120** may further include a browsing agent **125** installed therein. The browsing agent **125** may be, but is not limited to, a mobile application, a virtual application, a web application, a native application, and the like. In certain configurations, the browsing agent **125** can be realized as an add-on or plug-in for a web browser. In other configurations, the browsing agent **125** is a web browser. The user device **120** may receive a user query or otherwise receive a request to display content (e.g., via the browsing agent **125**) and to send, to the visual representation generator **130**, a request to generate a visual representation of the content to be browsed. The request to generate a visual representation may include, but is not limited to, the user query, the content to be browsed, an identifier of the content to be browsed, or a combination thereof. The user query may include a text query or a voice query. The user query may be submitted through a user gesture, e.g., tapping on a certain image or key word.

[0027] In an embodiment, the visual representation generator **130** is configured to receive the request to generate a visual representation and to determine an initial focal point based on the request. The initial focal point includes content to be initially displayed prominently (e.g., before navigation) to the user. Non-limiting examples of prominently displaying the initial focal point include displaying the initial focal point as larger than other content; displaying the initial focal point in a center, top, or other portion of a display; displaying the focal point with at least one prominence marker (e.g., a letter, a number, a symbol, a graphic, a color, etc.); displaying the focal point with a higher brightness or resolution than other content; displaying the focal point using one or more animations (e.g., displaying the focal point as moving up and down); a combination thereof; and the like. For example, if the content to be browsed includes images of a dog, a most recent image of dog may be selected as the initial focal point such that, when the visual representation is initially displayed to the user, the image of the dog is the largest and centermost image appearing on a display (not shown) of the user device **120**.

[0028] In a further embodiment, determining an initial focal point based on the request may further include preprocessing the user query. Pre-processing the user query may include, but is not limited to, correcting typos, enriching the query with information related to the user (e.g., a browsing history, a current location, etc.), and so on. In another embodiment, the initial focal point may include a web site utilized as a seed for a search. As an example, the initial focal point for a search based on the user query “buy shoes” may be a web site featuring a large variety of shoes.

[0029] In an embodiment, the visual representation generator **130** is configured to retrieve content from the retrieval systems **140** based on a focal point. For the first time content is retrieved for the request, the initial focal point is used. The retrieval systems **140** may search using the user query with respect to the focal point. Alternatively or collectively, the visual representation generator **130** may crawl through one or more of the retrieval systems **140** for the content. The retrieved content may include, but is not limited to, search results, content to be displayed, or both.

[0030] In another embodiment, the visual representation generator **130** may be configured to query an inventory management system **150** and to receive, from the inventory management system **150**, a probability that one or more vendors have a sufficient inventory of a product based on the user query and the focal point. An example implementation for an inventory management system for returning probabilities that vendors have sufficient inventories of product are described further in the above-referenced U.S. patent application Ser. No. 14/940,396 filed on Nov. 13, 2015, assigned to the common assignee, which is hereby incorporated by reference for all that it contains.

[0031] In an embodiment, the visual representation generator **130** is further configured to organize the retrieved content. The content may be organized around the focal point. Accordingly, the content, when initially organized, may be organized around the initial focal point. The visual representation generator **130** may be configured to receive user interactions respective of the organized content and to determine a current focal point based on the user interactions. In an embodiment, the content may be organized as points on a sphere and displayed to the user. The sphere may be displayed in a three-dimensional (3D) plane (i.e., using a stereoscopic display) or in a two-dimensional (2D) plane (i.e., such that the sphere appears to be 3D merely via optical illusion).

[0032] In another embodiment, the visual representation generator **130** is configured to generate a visual representation including a browsing environment and a plurality of dynamic content elements. Each dynamic content element includes content or representations of content to be browsed. In an embodiment, the focal point includes one of the dynamic content elements.

[0033] The browsing environment may include, but is not limited to, images, videos, or other visual illustrations of an environment in which the content is to be browsed. The visual illustrations may be two-dimensional, three-dimensional, and the like. The browsing environment may include visual illustrations of a real location (e.g., a store or other physical location), of a non-real location (e.g., a cartoon library, a virtual store, an imaginary combination of real stores, or any other virtual or fictional location), or any other visual illustrations (for example, a visual illustration showing text, objects, people, animals, solid colors, patterns, combinations thereof, and the like).

[0034] In an embodiment, the browsing environment is rendered at the beginning of browsing and static (i.e., remain the same as content is browsed) or dynamic (i.e., re-rendered or otherwise updated as content is browsed). As a non-limiting example, the browsing environment may include images showing a physical store in which products represented by the dynamic content elements are sold, where the images are updated to show different areas in the physical store as the user “moves” through the store by navigating among dynamic elements representing products sold in the store. As another non-limiting example, the browsing environment may include a static image illustrating a library, where the static image remains constant as the user navigates among dynamic elements representing books in the library.

[0035] The dynamic content elements may be updated and rendered in real-time as the user browses. Updating the dynamic content elements may include, but is not limited to, changing the content to be displayed, updating information related to each content (e.g., updating a value for a number of items in stock when the content represents a purchasable product), dynamically organizing the dynamic content elements, a combination thereof, and the like. Dynamic organization of the dynamic content elements may be based on one or more dynamic organization rules. Such dynamic organization rules may be based on, but are not limited to, amount of inventory in stock for store products (e.g., a current inventory or projected future inventory), popularity of content (e.g., content which is trending may be organized closer to the focal point), relevance to user interests (e.g., content that is more relevant to current user interests may be organized closer to the focal point), combinations thereof, and the like.

[0036] As a non-limiting example for a visual representation including a browsing environment and a plurality of dynamic content elements, the visual representation may represent an online store, with the browsing environment showing a storefront image and the dynamic content elements including product listings. The product listings may include, but are not limited to, text (e.g., product descriptions, product information such as inventory and price, etc.), images (e.g., images of the product), videos (e.g., videos demonstrating the product), sound (e.g., sound including customer reviews), combinations thereof, and the like. The dynamic content elements are rendered in real-time as the user browses. The rendered dynamic content elements include information related to the product listings, where the information includes at least a current or projected inventory. The rendered dynamic content elements are organized in real-time based on dynamic organization rules. The dynamic organization rules are based on inventory such that lower inventory items or items having minimal inventory (e.g., having an amount of inventory below a predetermined threshold) are organized closer to the focal point. Such organization may be useful for, e.g., incentivizing users to buy lower stock products. As the user browses, inventory information for the product listings is updated and the dynamic content elements are organized accordingly.

[0037] FIG. **3** shows an example screenshot illustrating a content sphere **300** which is a spherical visual representation of content. The content sphere **300** is organized around a focal point **310**. A plurality of images **320** act as points on the sphere representing content. If the focal point changes, the sphere may be rotated to show a different icon as the focal point. A horizontal axis **330** and a vertical axis **340** visually represent potential directions for changing the focal point to view additional content. For example, the user may gesture horizontally to view content oriented along the horizontal axis **330** and may gesture vertically to view content oriented along the vertical axis **340**. It should be note that the images **320** may include icons, textual information, widgets, or any other representation or presentation of the displayed content.

[0038] The axes **330** and **340** may be adaptably changed as the user selects new content to be the focal point **310** (e.g., by providing user gestures with respect to one of the images **320**), as the user rotates the content sphere **300** (e.g., by providing user gestures with respect to the axes **330** and **340**), or both. That is, the visual representation generator **130** is configured to predict (through a learning process), the user's path as the user browses via the presented content sphere **300**. As an example, if the focal point **310** includes content related to President Bill Clinton, then rotating the content sphere **300** down along the vertical axis **340** may return results related to the US in the 1990's. On the other hand, rotating the content sphere **310** in the right direction along the horizontal axis **330** may return results related to the Democratic party.

[0039] Further, as the content of the focal point **310** changes, the related content available via the axes **330** and **340** may change. Thus, axes of interest may be initially predefined, and then adaptably modified. For example, when the focal point **310** is changed to content related to President Obama by rotating the content sphere **300** along the horizontal axis **330**, the content available by rotating along the vertical access **340** may become content related to the US in the 2000's.

[0040] It should be appreciated that, by adaptably changing the content along the axes of interest (e.g., the axes **330** and **340**), the user may be provided with an endless browsing experience. Further, each content item can be presented in different virtual settings. As an example, a lipstick may be presented in the context of cosmetics and then again as part of a Halloween costume, thus continually experiencing new browsing experience.

[0041] In a further embodiment, the display may include a graphical user interface for receiving user interactions with respect to the spherically organized search results. As a non-limiting example, the search results for the query "alcoholic drinks" may be displayed as points on the content sphere **300** based on an initial focal point of a website featuring beer. Results from the initial focal point may be initially displayed as the focal point **310**. When a user rotates (e.g., swipes or moves) a mouse icon horizontally across the sphere **300**, a new focal point **310** may be determined as a web site for another type of alcoholic beverage (e.g., wine, vodka, and so on). When a user swipes or moves a mouse icon vertically across the content sphere **300**, a new focal point may be determined as a web site for a particular brand of beer.

[0042] It should be noted that the example content sphere **300** shown in FIG. **3** is merely an example of a visual representation and is not limiting on the disclosed embodiments. In particular, the content sphere **300** is shown as having two axes merely for illustrative purposes. Different numbers of axes may be equally utilized, any of which may be, but are not necessarily, horizontal or vertical. For example, diagonal axes may be utilized in addition to or instead of horizontal and vertical axes. Further, the axes may be three-dimensional without departing from the scope of the disclosure. For example, the content sphere **300** may be navigated by moving closer or farther away from a center point of the sphere.

[0043] It should further be noted that the content may be shown in a shape other than a spherical shape without departing from the scope of the disclosure. It should also be noted that the content sphere **300** is shown as having a solid black background surrounding the images **320** merely as an example illustration. Other browsing environments (e.g., other colors, patterns, static or dynamic images, videos, combinations thereof, etc.) may be equally utilized without departing from the scope of the disclosed environments. For example, the content may be rendered as three-dimensional representations of shelves and aisles of a real store, where the view of the shelves and aisles is updated as the user browses through images of products in the store.

[0044] FIG. **2** is an example flowchart **200** illustrating a method for refining search results according to an embodiment. In an embodiment, the method may be performed by a visual representation generator (e.g., the visual representation generator **130**, FIG. **1**). The method may be utilized to adaptively update visual representations of search results (e.g., search results displayed as the content sphere **300**, FIG. **3**).

[0045] At S**210**, a query by a user of a user device is received. The query may be received in the form of text, multimedia content, and so on. The query may be a textual query or a voice query.

[0046] At optional S**215**, the query may be preprocessed by, e.g., correcting typos, enriching the query with user information, and so on.

[0047] At S**220**, a focal point is determined based on the received query. The focal point may include a web site to be utilized as a seed for a search based on the query. The determination may include identifying one or more web sites related to the user query. The seed web site may be selected from among the identified web sites based on, e.g., relative validity of the sites (e.g., numbers of legitimate clicks or presence of malware). For example, a user query for "cheese" may result in identification of web sites related to grocery stores, restaurants, and so on. The seed website may be utilized as the initial focal point for the query such that content related to the seed website is displayed as the focal point prior to user interactions with respect to the visual representation.

[0048] At S**230**, at least one retrieval system is queried with respect to the received user query to retrieve search results. The focal point is further sent to the retrieval systems as a seed for the search. The retrieval systems may include, but are not limited to, search engines, inventory management systems, and other systems capable of retrieving content respective of queries.

[0049] In an embodiment, S**230** may further include querying at least one inventory management system for probabilities that products indicated in the search results are in stock at particular merchants. The probabilities may be utilized to, e.g., enrich or otherwise provide more information related to the search results. As a non-limiting example, the probability that a brand of shoe is in stock at a particular merchant may be provided in, e.g., a top left corner of an icon representing content related to the brand of shoe sold by the merchant.

[0050] At S**240**, a visual representation is generated based on the search results and the identified focal point. The visual representation may include points representing particular search results (e.g., a particular web page or a portion thereof). The visual representation may include a graphical user interface for receiving user interactions respective of the search results. In an embodiment, S**240** includes organizing the search results respective of the focal point. In a further embodiment, the search results may be organized graphically. In yet a further embodiment, the organization may include assigning the search results to points on, e.g., a sphere or other geometrical organization of the search results.

[0051] In another embodiment, the generated visual representation may include a browsing environment and a plurality of dynamic content elements. The browsing environment may include, but is not limited to, images, videos, or other visual illustrations of an environment (e.g., a real location, a virtual location, background colors and patterns, etc.) in which content is to be browsed. The browsing environment may be static, or may be dynamically updated in real-time as a user browses the content. The dynamic content elements are updated in real-time as a user browses the content. The dynamic content elements may be further reorganized in real-time based on the user browsing. As a non-limiting example, the generated visual representation may include a static storefront image and a plurality of dynamic elements updated in real-time to show product listings with current inventories, where the dynamic elements are organized such that lower inventory content items are closer to the focal point that higher inventory items.

[0052] At S**250**, the visual representation of the organized search results is caused to be displayed to the user. In an embodiment, S**250** may include sending the visual representation to a user device (e.g., the user device **120**, FIG. **1**). In an embodiment, the visual representation may be displayed as a three-dimensional (3D) representation of the search results.

[0053] At S**260**, at least one user input is received with respect to the displayed visual representation. The user inputs may include, but are not limited to, key strokes, mouse clicks, mouse movements, user gestures on a touch screen (e.g., tapping, swiping), movement of a user device (as detected by, e.g., an accelerometer, a global positioning system (GPS), a gyroscope, etc.), voice commands, and the like.

[0054] At S**270**, based on the received user inputs, the search results are refined. In an embodiment, S**270** may include determining a current focal point based on the user inputs and the visual representation. In a further embodiment, S**270** includes updating the visual representation using a website of the new focal point as a seed for the search.

[0055] At S**280**, it is determined whether additional user inputs have been received and, if so, execution continues with S**260**; otherwise, execution terminates. In another embodiment, S**280** includes determining if the additional user inputs include a new or modified query and, if so, execution may continue with S**210**.

[0056] FIG. **4** is an example flowchart **400** illustrating a method for displaying content for intuitive browsing according to an embodiment. In an embodiment, the method may be performed by a visual representation generator (e.g., the visual representation generator **130**). The visual representation generator may query, crawl, or otherwise obtain content from content retrieval systems (e.g., the content retrieval systems **140**). In another embodiment, the method may be performed by a user device (e.g., the user device **120**) based on locally available content, retrieved content, or both.

[0057] At S**410**, a request to display content is received. The request may include, but is not limited to, a query, content to be displayed, an identifier of content to be displayed, an identifier of at least one source of content, a combination thereof, and so on.

[0058] At S**420**, based on the request, a focal point is determined. The focal point may be determined based on, but not limited to, the query, the content to be displayed, the designated content sources, information about a user (e.g., a user profile, a browsing history, demographic information, etc.), combinations thereof, and so on. The focal point may be, but is not limited to, content, a source of content, a category or other grouping of content, a representation thereof, and so on. In an embodiment, the focal point may be related to a website to be used as a seed for a search with respect to the query (e.g., a web crawl).

[0059] At S**430**, content to be browsed with respect to the focal point is identified. The identified content may be related to the focal point. The identified content may be stored locally, or may be retrieved from at least one data source (e.g., the content retrieval systems **140** or the inventory management system **150**, FIG. **1**) As examples, the identified content may include, but is not limited to, content from the same or similar web sources, content that is contextually related to content of the focal point (e.g., belonging to a same category or otherwise sharing common or related information), and so on. Similarity of content may be based on matching the content. In an embodiment, content and sources may be similar if they match above a predefined threshold.

[0060] At S**440**, the identified content is organized with respect to the focal point. In an embodiment, the organization may be based on one or more axes. The axes may represent different facets of the determined content such as, but not limited to, creator (e.g., an artist, author, director, editor, publisher, etc.), geographic location, category of subject matter, type of content, genre, time of publication, and any other point of similarity among content. As a non-limiting example, content related to a focal point of a particular movie may be organized based on one or more axes such as, but not limited to, movies featuring the same actor(s), movies by the same director, movies by the same publisher, movies within the same genre, movies originating in the same country, movies from a particular decade or year, other media related to the movie (e.g., a television show tying into the movie) merchandise or other products related to the movie, and so on.

[0061] At S**450**, a visual representation of the organized content is generated. The visual representation may include points, each point representing at least a portion of the identified content. The visual representation may include a graphical user interface for receiving user interactions respective of the search results. In an embodiment, the visual representation may be spherical, may allow a user to change axes by gesturing horizontally, and may allow a user to change content within an axis by gesturing vertically. In another embodiment, the visual representation may be three-dimensional.

[0062] At S**460**, the generated visual representation is caused to be displayed on a user device. In an embodiment, S**460** includes sending the visual representation to the user device. In an embodiment, the visual representation may be updated when, e.g., an amount of content that has been displayed is above a predefined threshold, a number of user interactions is above a predefined threshold, a refined or new query is received, and the like.

[0063] As a non-limiting example, a request to display content is received. The request includes the query “the thinker.” A focal point including an image of the sculpture “The Thinker” by Auguste Rodin is determined. Content related to “The Thinker,” including various sculptural and artistic works, is determined using the website in which the image is shown as a seed for a search. The content is organized spherically based on axes including other famous sculptures, sculptures by French sculptors, art by Auguste Rodin, works featuring “The Thinker,” sculptures created in the late 1800s, and versions of “The Thinker” made from different materials. A visual representation of the spherically organized content is generated and caused to be displayed on a user device.

[0064] FIG. **5** is an example schematic diagram of the visual representation generator **130** according to an embodiment. The visual representation generator **130** includes a processing circuitry **510** coupled to a memory **515**, a storage **520**, and a network interface **530**. In another embodiment, the components of the visual representation generator **130** may be communicatively connected via a bus **540**.

[0065] The processing circuitry **510** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

[0066] The memory **515** may be volatile (e.g., RAM, etc.), non-volatile (e.g., ROM, flash memory, etc.), or a combination thereof. In one configuration, computer readable instructions to implement one or more embodiments disclosed herein may be stored in the storage **520**.

[0067] In another embodiment, the memory **515** is configured to store software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the one or more processors, cause the processing circuitry **510** to perform the various processes described herein. Specifically, the instructions, when executed, cause the processing circuitry **510** to perform generation of visual representations of content for intuitive browsing, as discussed hereinabove.

[0068] The storage **520** may be magnetic storage, optical storage, and the like, and may be realized, for example, as flash memory or other memory technology, CD-ROM, Digital Versatile Disks (DVDs), or any other medium which can be used to store the desired information.

[0069] The network interface **530** allows the visual representation generator **130** to communicate with the user device **120**, the content retrieval systems **140**, the inventory management system **150**, or a combination of, for the purpose of, for example, obtaining requests, obtaining content, obtaining probabilities, querying, sending visual representations, combinations thereof, and the like.

[0070] It should be understood that the embodiments described herein are not limited to the specific architecture illustrated in FIG. **5**, and other architectures may be equally used without departing from the scope of the disclosed embodiments.

[0071] It should be noted that various embodiments described herein are discussed with respect to content from search results merely for simplicity purposes and without limitation on the disclosed embodiments. Other content, including content preexisting on a device, content available via a storage or data source, and the like, may be displayed and browsed without departing from the scope of the disclosure.

[0072] It should be further noted that the embodiments described herein are discussed with respect to a spherical representation of content merely for simplicity purposes and without limitation on the disclosed embodiments. Other geometrical representations may be utilized with points on the geometric figures representing search results without departing from the scope of the disclosure. For example, the search results may be organized as points on different sides a cube such that user interactions may cause the displayed cube side to change, thereby changing the search results being displayed.

[0073] It should also be noted that various examples for changing content are provided merely for the sake of illustration and without limitation on the disclosed embodiments. Content may be organized in other ways without departing from the scope of the disclosure.

[0074] It should be further noted that the content may be organized based on the subject matter of the content. For example, the content may be organized differently for queries for restaurants than for requests to display documents on a user device.

[0075] It should be understood that any reference to an element herein using a designation such as "first," "second," and so forth does not generally limit the quantity or order of those elements. Rather, these designations are generally used herein as a convenient method of distinguishing between two or more elements or instances of an element. Thus, a reference to first and second elements does not mean that only two elements may be employed there or that the first element must precede the second element in some manner. Also, unless stated otherwise, a set of elements comprises one or more elements.

[0076] As used herein, the phrase "at least one of" followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including "at least one of A, B, and C," the system can include A alone; B alone; C alone; A and B in combination; B and C in combination; A and C in combination; or A, B, and C in combination.

[0077] The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPUs"), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

[0078] All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

What is claimed is:

1. A method for intuitive content browsing, comprising:

determining, based on a request to browse content, an initial focal point for a visual representation of the content, wherein the initial focal point represents a content item;

identifying, based on the request and the determined initial focal point, the content to be browsed;

generating, based on the identified content and the focal point, a visual representation of the identified content, wherein the generated visual representation includes the identified content organized with respect to the initial focal point; and

sending, to a user device, the generated visual representation for display, wherein the identified content is browsed via the displayed visual representation with respect to the focal point.

2. The method of claim **1**, wherein the generated visual representation includes at least one axis, wherein the generated visual representation is browsed along each of the at least one axis.

3. The method of claim **2**, wherein the browsing of the displayed visual representation includes selecting, based on at least one user input, a new focal point, wherein the displayed visual representation is updated with respect to the new focal point.

4. The method of claim **1**, wherein the request includes a query, wherein the content to be browsed includes at least search results.

5. The method of claim **4**, wherein the determined focal point includes content of a web site, wherein identifying the content to be browsed further comprises:

searching, based on the query, in at least one content retrieval system for the content to be browsed, wherein the web site is utilized as a seed for the search.

6. The method of claim **4**, further comprising:

querying at least one inventory management system for probabilities that products indicated in the search results are available from at least one merchant.

7. The method of claim **4**, further comprising:

determining, based on at least one user input, a new focal point, the new focal point including content of a web site; and

updating the visual representation based on the new focal point.

8. The method of claim **1**, wherein the generated visual representation includes a browsing environment and at least one dynamic content element, the browsing environment including at least one visual illustration, each dynamic content element including one of the content to be browsed, wherein the at least one dynamic content element is updated in real-time as the identified content is browsed.

9. The method of claim **8**, wherein the browsing environment is updated in real-time as the identified content is browsed.

10. A non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to perform a process, the process comprising:

determining, based on a request to browse content, an initial focal point for a visual representation of the content, wherein the initial focal point represents a content item;

identifying, based on the request and the determined initial focal point, the content to be browsed;

generating, based on the identified content and the focal point, a visual representation of the identified content, wherein the generated visual representation includes the identified content organized with respect to the initial focal point; and

sending, to a user device, the generated visual representation for display, wherein the identified content is browsed via the displayed visual representation with respect to the focal point.

11. A system for intuitive content browsing, comprising:

a processing circuitry; and

a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to:

determine, based on a request to browse content, an initial focal point for a visual representation of the content, wherein the initial focal point represents a content item;

identify, based on the request and the determined initial focal point, the content to be browsed;

generate, based on the identified content and the focal point, a visual representation of the identified content, wherein the generated visual representation includes the identified content organized with respect to the initial focal point; and

send, to a user device, the generated visual representation for display, wherein the identified content is browsed via the displayed visual representation with respect to the focal point.

12. The system of claim **11**, wherein the generated visual representation includes at least one axis, wherein the generated visual representation is browsed along each of the at least one axis.

13. The system of claim **12**, wherein the browsing of the displayed visual representation includes selecting, based on at least one user input, a new focal point, wherein the displayed visual representation is updated with respect to the new focal point.

14. The system of claim **11**, wherein the request includes a query, wherein the content to be browsed includes at least search results.

15. The system of claim **14**, wherein the determined focal point includes content of a web site, wherein the system is further configured to:

search, based on the query, in at least one content retrieval system for the content to be browsed, wherein the web site is utilized as a seed for the search.

16. The system of claim **14**, wherein the system is further configured to:

query at least one inventory management system for probabilities that products indicated in the search results are available from at least one merchant.

17. The system of claim **14**, wherein the system is further configured to:

determine, based on at least one user input, a new focal point, the new focal point including content of a web site; and

update the visual representation based on the new focal point.

18. The system of claim **11**, wherein the generated visual representation includes a browsing environment and at least one dynamic content element, the browsing environment including at least one visual illustration, each dynamic content element including one of the content to be browsed, wherein the at least one dynamic content element is updated in real-time as the identified content is browsed.

19. The system of claim **18**, wherein the browsing environment is updated in real-time as the identified content is browsed.

* * * * *

EXHIBIT E

CLAIMS

What is claimed is:

1. A method for securing virtual cloud assets at rest against cyber threats, comprising:

determining a location of a view of at least one virtual disk of a protected virtual cloud asset, wherein the virtual cloud asset is at rest and, when activated, instantiated in the cloud computing environment;

accessing the view of the virtual disk based on the determined location;

analyzing the view of the protected virtual cloud asset to detect potential cyber threats risking the protected virtual cloud asset, wherein the virtual cloud asset is inactive during the analysis; and

alerting detected potential cyber threats based on a determined priority.

2. The method of claim 1, further comprising:

prioritizing each detection of potential cyber threats based on their respective risk to the protected virtual cloud asset; and

mitigating a potential cyber threat posing a risk to the protected virtual cloud asset.

3. The method of claim 1, wherein determining the location of the view of at least one virtual disk further comprises:

determining a virtual disk allocated to the protected virtual cloud asset.

4. The method of claim 3, further comprising:

querying a cloud management console of the cloud computing platform to determine the location of the view and the location of the virtual disk.

5. The method of claim 1, further comprising:

requesting the cloud computing platform to create a new view of the protected virtual cloud asset, when an existing snapshot cannot be located.

6. The method of claim 1, wherein analyzing the view of the protected virtual cloud asset further comprises:

parsing a copy of the view; and

scanning the parsed copy to detect the potential cyber threats, wherein the potential cyber threats include known and unknown vulnerabilities, and wherein the detection is based on a type of vulnerability.

7. The method of claim 6, wherein scanning the parsed copy further comprises any one of:

checking configuration files of each application, operating system, and system installed in the protected virtual cloud asset;

verifying access times to files by the operating system installed in the operating system;

analyzing system logs to deduce what applications and modules executed in the protected virtual cloud asset; and

analyzing machine memory stored in the view to deduce what applications and modules executed in the protected virtual cloud asset.

8. The method of claim 6, further comprising:

instantiating an instance copy of the protected virtual cloud asset from the view; and

monitoring all activity performed by the instance of the protected virtual cloud asset.

9. The method of claim 1, wherein the protected virtual cloud asset includes any one of: a virtual machine, a software container, a micro-service.

10. A non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to execute a process for securing virtual cloud assets at rest against cyber threats, the process comprising:

determining a location of a view of at least one virtual disk of a protected virtual cloud asset, wherein the virtual cloud asset is at rest and, when activated, instantiated in a cloud computing environment;

accessing the view of the virtual disk based on the determined location;

analyzing the view of the protected virtual cloud asset to detect potential cyber threats risking the protected virtual cloud asset, wherein the virtual cloud asset is inactive during the analysis; and

alerting detected potential cyber threats based on a determined priority.

11. A system for securing virtual cloud assets at rest against cyber threats, comprising:

a processing circuitry; and

a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to:

determine a location of a view of at least one virtual disk of a protected virtual cloud asset, wherein the virtual cloud asset is at rest and, when activated, instantiated in a cloud computing environment;

access the view of the virtual disk based on the determined location;

analyze the view of the protected virtual cloud asset to detect potential cyber threats risking the protected virtual cloud asset, wherein the virtual cloud asset is inactive during the analysis; and

alert detected potential cyber threats based on a determined priority.

12. The system of claim 11, wherein the system is further configured to:

prioritize each detection of potential cyber threats based on their respective risk to the protected virtual cloud asset; and

mitigate a potential cyber threat posing a risk to the protected virtual cloud asset.

13. The system of claim 11, wherein determining the location of the view of at least one virtual disk further comprises:

determining a virtual disk allocated to the protected virtual cloud asset.

14. The system of claim 13, wherein the system is further configured to:

query a cloud management console of the cloud computing platform to determine the location of the view and the location of the virtual disk.

15. The system of claim 11, wherein the system is further configured to:

requesting the cloud computing platform to create a new view of the protected virtual cloud asset, when an existing snapshot cannot be located.

16. The system of claim 11, wherein analyzing the view of the protected virtual cloud asset further comprises:

parsing a copy of the view; and

scanning the parsed copy to detect the potential cyber threats, wherein the potential cyber threats include known and unknown vulnerabilities, and wherein the detection is based on a type of vulnerability.

17. The system of claim 16, wherein scanning the parsed copy further comprises any one of:

checking configuration files of each application, operating system, and system installed in the protected virtual cloud asset;

verifying access times to files by the operating system installed in the operating system;

analyzing system logs to deduce what applications and modules executed in the protected virtual cloud asset; and

analyzing machine memory stored in the view to deduce what applications and modules executed in the protected virtual cloud asset.

18. The system of claim 16, wherein the system is further configured to:

instantiate an instance copy of the protected virtual cloud asset from the view; and

monitor all activity performed by the instance of the protected virtual cloud asset.

19. The system of claim 11, wherein the protected virtual cloud asset includes any one of: a virtual machine, a software container, a micro-service.

EXHIBIT F

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

BSD CROWN, LTD.,
Plaintiff,
v.
AMAZON.COM, INC., et al.,
Defendants.

Case No. 3:23-cv-00057-WHO

ORDER ON MOTION TO DISMISS

Re: Dkt. No. 32

United States District Court
Northern District of California

Plaintiff BSD Crown, LTD ("BSD") filed this case alleging direct and willful infringement of its patent, U.S. No. 6,389,473 ("'473 Patent"). Defendants Amazon.com, Inc. ("Amazon.com"), Amazon Web Services, Inc. ("AWS") and Twitch Interactive, Inc. ("Twitch") (and collectively, "defendants") filed a motion to dismiss. For the reasons explained below, the motion is GRANTED in part and DENIED in part, with leave to amend.

BACKGROUND

I. FACTUAL BACKGROUND

The United States Patent and Trademark Office ("USPTO") issued the relevant patent-in-suit, the '473 Patent, on May 14, 2002. Complaint ("Compl.") [Dkt. No. 1] ¶ 21. BSD alleges that the '473 Patent gives BSD exclusive rights to a video streaming technology called "Dynamic Adaptive Streaming over HTTP" and "HTTP Live Streaming." *Id.* ¶¶ 2, 6, 57–71. It explains that the patented technology improves real-time video streaming broadcasts. *Id.* ¶¶ 2, 23. It alleges that Amazon.com, AWS, and Twitch currently use this technology to stream live broadcasts and consequently infringe on BSD's patent. *Id.* ¶¶ 57–64.

BSD says that Amazon.com used Baker & Hostetler LLP of Philadelphia as its patent prosecution counsel and that the firm learned of the '473 Patent while prosecuting Amazon.com's

United States District Court
Northern District of California

patent because the ’473 Patent was included as a primary prior art reference in an issued rejection. *Id.* ¶¶ 42–46. It asserts that Amazon.com “solicited an interview with the patent examiner” after its patent application was initially rejected by USPTO. *Id.* ¶ 47. BSD alleges that Amazon.com’s proposed amendments to its patent application and its interview summary from February 11, 2016, which both reference the ’473 Patent as alleged prior art, demonstrate its knowledge of the patent-in-suit. *Id.* ¶ 47; Ex. 12. at 42–44. According to BSD, parent company and defendant Amazon.com learned of the ’473 Patent in 2015 or earlier through the USPTO patent prosecution process, when one of Amazon.com’s patent applications was rejected. *See id.* ¶¶ 40–51.

BSD alleges that AWS and Twitch also used Baker & Hostetler and that the firm imputed knowledge of the ’473 Patent to AWS and Twitch. *Id.* ¶¶ 52-55, 74, 85. It notes the parent-subsidiary relationship of Amazon.com to AWS and Twitch, *id.* ¶¶ 69–76, and points to USPTO records that show that “at least three” of Twitch’s patents were prosecuted under Amazon.com’s Baker & Hostetler account rather than under an account assigned solely to Twitch, *id.* ¶ 53; Ex. 13.

BSD brings one cause of action for patent infringement against Amazon.com and AWS. *Id.* ¶¶ 77-81. It also brings one cause of action for patent infringement against Twitch. *Id.* ¶¶ 82-86. Both causes of action are premised on claims of direct infringement or infringement under the doctrine of equivalents. *Id.* ¶¶ 78, 83. BSD alleges that all defendants acted egregiously because they knew or were willfully blind to their infringing acts. *Id.* ¶ 56.

II. PROCEDURAL BACKGROUND

The defendants filed a joint motion to dismiss the claims of willful infringement. (“Mot.”) [Dkt. No. 32]. BSD opposed. (“Oppo.”) [Dkt. No. 40]. The defendants replied. (“Repl.”) [Dkt. No. 41]. I held a hearing at which counsel for both parties appeared.

LEGAL STANDARD

I. RULE 8(A)(2)

A complaint must contain “a short and plain statement of the claim showing that the pleader is entitled to relief.” Fed. R. Civ. Proc. 8(a)(2). “[T]he ‘short and plain statement’ must provide the defendant with ‘fair notice of what the plaintiff’s claim is and the grounds upon which it rests.'" *Dura Pharms., Inc. v. Broudo*, 544 U.S. 336, 346 (2005) (quoting *Conley v. Gibson*, 355 U.S. 41, 47 (1957)). To comply with Rule 8(a), a plaintiff "must allege the basis of [its] claim against each defendant . . . to put defendants on sufficient notice of the allegations against them." *Gauvin v. Trombatore*, 682 F. Supp. 1067, 1071 (N.D. Cal. 1988).

II. RULE 12(B)(6)

Under Federal Rule of Civil Procedure 12(b)(6), a district court must dismiss a complaint if it fails to state a claim upon which relief can be granted. To survive a Rule 12(b)(6) motion to dismiss, the plaintiff must allege "enough facts to state a claim to relief that is plausible on its face." *Bell Atl. Corp. v. Twombly*, 550 U.S. 544, 570 (2007). A claim is facially plausible when the plaintiff pleads facts that "allow the court to draw the reasonable inference that the defendant is liable for the misconduct alleged." *Ashcroft v. Iqbal*, 556 U.S. 662, 678 (2009) (citation omitted). There must be "more than a sheer possibility that a defendant has acted unlawfully." *Id.* While courts do not require "heightened fact pleading of specifics," a plaintiff must allege facts sufficient to "raise a right to relief above the speculative level." *Twombly*, 550 U.S. at 555, 570.

In deciding whether the plaintiff has stated a claim upon which relief can be granted, the court accepts the plaintiff's allegations as true and draws all reasonable inferences in favor of the plaintiff. *See Usher v. City of Los Angeles*, 828 F.2d 556, 561 (9th Cir. 1987). However, the court is not required to accept as true "allegations that are merely conclusory, unwarranted deductions of fact, or unreasonable inferences." *In re Gilead Scis. Sec. Litig.*, 536 F.3d 1049, 1055 (9th Cir. 2008).

If the court dismisses the complaint, it "should grant leave to amend even if no request to amend the pleading was made, unless it determines that the pleading could not possibly be cured by the allegation of other facts." *Lopez v. Smith*, 203 F.3d 1122, 1127 (9th Cir. 2000). In making this determination, the court should consider factors such as "the presence or absence of undue delay, bad faith, dilatory motive, repeated failure to cure deficiencies by previous amendments, undue prejudice to the opposing party and futility of the proposed amendment." *Moore v. Kayport Package Express*, 885 F.2d 531, 538 (9th Cir. 1989).

United States District Court
Northern District of California

DISCUSSION

The defendants move to dismiss BSD's willful infringement claims, arguing that BSD failed to plead (1) that AWS and Twitch had the requisite knowledge of the patent, (2) that any defendant had the specific intent to infringe, and (3) that any defendant engaged in egregious conduct. *See* Mot. BSD contests each argument. *See* Oppo.

To prove a willful infringement claim, a jury must find that the defendant had pre-suit knowledge of the patent-in-suit and that the defendant infringed deliberately or intentionally. *See, e.g., Eko Brands, LLC v. Adrian Rivera Maynez Enters., Inc.*, 946 F.3d 1367, 1378 (Fed. Cir. 2020) (citing *Halo Elecs., Inc. v. Pulse Elecs., Inc.*, 579 U.S. 93, 105–06 (2016)); *SRI Int'l, Inc. v. Cisco Sys., Inc.*, 14 F.4th 1323, 1329–30 (Fed. Cir. 2021); *BASF Plant Sci., LP v. Commonwealth Sci. & Indus. Rsch. Org.*, 28 F.4th 1247, 1274 (Fed. Cir. 2022). In other words, "[t]o establish willfulness, the patentee must show the accused infringer had [the] specific intent to infringe at the time of the challenged conduct." *Bayer Healthcare LLC v. Baxalta Inc.*, 989 F.3d 964, 987–88 (Fed. Cir. 2021) (citing *Halo*, 579 U.S. at 105–06).

However, no binding appellate decisions discuss the pleading requirements for a willful infringement claim. *See Sonos, Inc. v. Google LLC*, 591 F. Supp. 3d 638, 643 (N.D. Cal. 2022) ("No post-*Halo* appellate authority addresses any pleading requirements for . . . willful infringement."). Rather, in applying the Federal Circuit's rules for *proving* willful infringement after *Eko*, courts in this district have reasoned that to survive a motion to dismiss, a plaintiff must plausibly plead "knowledge of the patent and knowledge of infringement." *Id.*; *see also Dali Wireless, Inc. v. Corning Optical Comm'ns LLC*, Case No. 20-6469-EMC, 2022 WL 16701926, at *3 (N.D. Cal. Nov. 3, 2022).

And courts in this district diverge in whether plaintiffs must plausibly plead that the defendants engaged in egregious conduct. *Compare Fortinet, Inc. v. Forescout Techs., Inc.*, 543 F. Supp. 3d 814, 840 (N.D. Cal. 2021) (requiring plaintiffs to plead egregiousness), *with Sonos*, 591 F. Supp. 3d at 647 (rejecting the requirement to plead egregiousness).

I. KNOWLEDGE OF THE PATENT

The defendants argue that BSD failed to plead pre-suit knowledge of the '473 Patent for

United States District Court
Northern District of California

AWS and Twitch, though they do not contest Amazon.com's knowledge of the patent. *See* Mot. at 5–7 & n.2; Repl. at 5–9. BSD asserts that the companies' patent prosecution counsel imputed knowledge of the '473 patent to AWS and Twitch through a principal-agent relationship and that the parent-subsidiary relationship between the companies helps establish knowledge.[1] Oppo. 7:1-5; 8:13-10:1; Compl. ¶¶ 52–53.

A plaintiff must plausibly plead pre-suit knowledge of the patent for willful infringement to survive a motion to dismiss. *Illumina, Inc. v. BGI Genomics Co., Ltd.*, No. 19-03770-WHO, 2020 WL 571030, at *6 (N.D. Cal. Feb. 5, 2020); *see also Sonos*, 591 F. Supp. 3d at 644; *Word to Info, Inc. v. Google, Inc.*, 140 F. Supp. 3d 986, 989–90 (N.D. Cal. 2015). BSD's theory of knowledge is that Baker & Hostetler imputed knowledge to AWS and Twitch, so first I turn to the standard for pleading knowledge imputation.

The Federal Circuit has acknowledged that both state law and federal common law may apply to questions of imputed knowledge. *See Long Island Savings Bank, FSB v. United States*, 503 F.3d 1234, 1249 (Fed. Cir. 2007) (reasoning that federal and New York law had "similar standards" for knowledge imputation, and analyzing the case under both standards); *see also Belton Indus., Inc. v. United States*, 6 F.3d 756, 761 (Fed. Cir. 1993) (applying federal common law for knowledge of notice); *Atasi Corp. v. Seagate Tech.*, 847 F.2d 826, 829–30 (Fed. Cir. 1988) (applying Ninth Circuit law, which applied California state law to determine whether the knowledge of an attorney with confidential information is imputed to the entire firm); *Immunocept, LLC v. Fulbright & Jaworski, LLP*, 504 F.3d 1281, 1287 (Fed. Cir. 2007) (applying Texas law to knowledge imputation analysis). Here, then, Ninth Circuit and state law precedent guide questions of knowledge imputation.

In the Ninth Circuit, "[c]lients are 'considered to have notice of all facts known to their lawyer-agent.'" *Cmty. Dental Servs. v. Tani*, 282 F.3d 1164, 1168 (9th Cir. 2002) (quoting *Ringgold Corp. v. Worrall*, 880 F.2d 1138, 1141–42 (9th Cir. 1989)). "Because the client is

[1] BSD also asserts that that Twitch is an alter ego of AWS, and that AWS uses Twitch technology, *see* Oppo. 7:6-17, but even if true, it is not clear how that shows either company knew of the patent. And, alleging that Amazon and Twitch have "deep ties" is too high-level to plausibly allege Twitch knew of the patent. *See id.* 7:18-22.

presumed to have voluntarily chosen the lawyer as his representative and agent, he ordinarily cannot later avoid accountability for negligent acts or omissions of his counsel." *Id.* (citing *Link v. Wabash R.R. Co.*, 370 U.S. 626, 633–34 (1962)). However, more recently in the bankruptcy context, the Ninth Circuit held that there is no precedent supporting "imput[ing] to a client knowledge that [its] lawyer gained while representing a *different* client." *In re Perle*, 725 F.3d 1023, 1028 (9th Cir. 2013). In the context of professional responsibility, a court in the Eastern District reasoned that imputation of an attorney's knowledge to an entire law firm may depend on whether that attorney was "personally involved" in the prior case. *See Dieter v. Regents of Univ. of Cal.*, 963 F. Supp. 908, 911 (E.D. Cal. Apr. 21,1997); *cf. Asyst Techs., Inc. v. Empak, Inc.*, 962 F. Supp. 1241, 1242 (N.D. Cal. Apr. 18, 1997) (reasoning that where an attorney with knowledge of confidential information "is currently a member of a law firm, her knowledge is imputed to all members of the firm" to disqualify the firm from representing adverse counsel). A district court in Washington also explained that an attorney's knowledge can be imputed to its client, but its reasoning indicated that imputation is relatively fact dependent. *See Glob. Enters., LLC v. Montgomery Purdue Blankenship & Austin PLLC*, 52 F. Supp. 3d 1162, 1168 (W.D. Wash. 2014), *aff'd*, 691 F. App'x 460 (9th Cir. 2017).

More recently, a court in the Central District of California contemplated an imputation theory of pre-suit knowledge in the context of patent prosecution for the purposes of a 12(b)(6) motion to dismiss. *See Core Optical Techs., LLC v. Nokia Corp.*, No. SA-cv-1902190-JAK-RAOX, 2020 WL 6126285 at *6–7 (C.D. Cal. Oct. 8, 2020) (hereinafter "*Nokia*"). In *Nokia*, like this case, the plaintiff argued that pre-suit knowledge of a patent originated from a prior art reference cited by the USPTO during patent prosecution. *Id.* at *6. They contended "that the knowledge of a party's patent prosecution lawyers can be imputed to [another] party." *Id.* But there, the court declined to find a plausible inference of pre-suit knowledge because plaintiff failed to specify which counsel actually prosecuted the patents at issue. *Id.*

In this case, whether knowledge is imputed from Baker & Hostetler to AWS and Twitch depends on several factors, including the relationship between Baker & Hostetler and Amazon.com, whether AWS and Twitch are "different clients" from Amazon.com, whether the

United States District Court
Northern District of California

individual counsel who supervised Amazon.com's patent prosecution also supervised AWS and Twitch's patents, and other relevant facts. *See In re Perle*, 725 F.3d at 1028; *Dieter*, 936 F. Supp. at 911; *Asyst Techs.*, 962 F. Supp. at 1242; *Glob. Enters.*, 52 F. Supp. 3d at 1168; *Nokia*, 2020 WL 6126285, at *7. To support its theory, BSD points to USPTO records that show that "at least three" of Twitch's patents were prosecuted under Amazon.com's Baker & Hostetler account rather than under an account assigned solely to Twitch, Compl. ¶ 53; Ex. 13, and points to the parent-subsidiary relationship of Amazon.com to AWS and Twitch, Compl. ¶¶ 69–76; Oppo. at 7.

Given the state of Ninth Circuit law and the low bar as explained by *Illumina*, it seems possible for a plaintiff to allege a theory of knowledge imputation by pleading that the defendant's counsel knew of the patent and so the defendant knew of it, too. *See Nokia*, 2020 WL 6126285, at *6–7; *Cmty. Dental Servs.*, 282 F.3d at 1168. Here, for example, BSD might plead that the same attorneys who learned of the patent while prosecuting Amazon.com's patent then worked on patent prosecution at Twitch and AWS—or even that the patent prosecution team at Baker & Hostetler is sufficiently small that it is plausible that the same attorneys worked on both cases—and so imputed their knowledge to Twitch and AWS in that work. *Cf. Dieter*, 936 F. Supp. at 911 (noting an attorney's personal involvement in cases affects knowledge imputation); *see also Asyst Techs.*, 962 F. Supp. at 1242. BSD could, for example, allege that Amazon.com, AWS, and Twitch are not "different client[s]" and so knowledge gained working for each is imputed to the others, *In re Perle*, 725 F.3d at 1028 (emphasis omitted). But the closest BSD gets to such an allegation is that three Twitch patents were prosecuted under a Baker & Hostetler account assigned to "Amazon," Compl. ¶¶ 52-53 (emphasis omitted), from which it would be a stretch to infer that the same attorneys who previously learned of the '473 Patent by prosecuting Amazon.com's patent also later conducted relevant work at Twitch. Combined with BSD's failure to engage with relevant knowledge imputation law, or even to point to case law addressing parent-subsidiary relationships and knowledge imputation, BSD does not plausibly allege that the same counsel knew of BSD's patent and imputed that knowledge to AWS and Twitch. Accordingly, even with the low bar and flexible test for knowledge, BSD's complaint does not plausibly allege knowledge for AWS or Twitch.

Lastly, and as relevant if BSD chooses to amend its complaint, I note that I am unpersuaded by the defendants' citation to *Olaf Soot Design, LLC v. Daktronics, Inc.*, 325 F. Supp. 3d 456, 464 (S.D.N.Y. 2018), for the proposition that a knowledge imputation theory in the context of patent prosecution has been "soundly rejected" in the courts. *See* Mot. at 6. In *Olaf*, the plaintiff alleged that knowledge from outside patent prosecution counsel was imputed to the defendant, but the court found knowledge was not imputed because evidence in the record negated that theory. 325 F. Supp. 3d at 463. For example, correspondence between patent prosecution counsel revealed that it advised the defendant-client that they did not need to review the USPTO's rejection and that "a response to [the] patent rejection would be drafted 'without requiring any work'" from the defendant-client. *Id.* Moreover, the defendant-client had no in-house counsel to review a counsel's work. *Id.* Here, though, neither BSD nor the defendants have pointed to similar facts as those in *Olaf* which would tend to prove or disprove a defendant's pre-suit knowledge. And, even if there were such facts, *Olaf* is not wholly applicable because it focused on a finding of willfulness, not on a finding of knowledge.

Accordingly, BSD fails to sufficiently plead pre-suit knowledge of the patent for AWS and Twitch. The claims against AWS and Twitch are DISMISSED with leave to amend.

II. SPECIFIC INTENT TO INFRINGE

The defendants also argue that BSD fails to plausibly plead that any defendant specifically intended to infringe. Mot. at 3:28–5:24. In opposition, BSD asserts that its allegations concerning Amazon.com's pre-suit knowledge of the patent combined with the "detailed claim charts" showing that Amazon.com's products infringe are sufficient to show that all defendants intended to infringe. *See* Oppo. 10:4-13:4.

"Knowledge of the asserted patent and evidence of infringement is necessary, but not sufficient, for a finding of willfulness. Rather, willfulness requires deliberate or intentional infringement." *Bayer Healthcare*, 989 F.3d at 988; *see also BASF Plant Sci*, 28 F.4th at 1274 (same). To plead deliberate or intentional infringement, plaintiffs may plead "knowledge of infringement," which they can do by pleading that "the accused infringer had a specific intent to infringe at the time of the challenged conduct." *Sonos*, 591 F. Supp. 3d at 643 (quoting *Bayer*

Healthcare, 989 F.3d at 987–88); *Dali Wireless*, 2022 WL 16701926, at *3 (same). Because knowledge of infringement is different from knowledge of the patent, distinct allegations must be pleaded in the complaint. *See Dali Wireless*, 2022 WL 16701926, at *6. And while knowledge of infringement must be pled with plausibility, *see Sonos*, 591 F. Supp. 3d at 643, plaintiffs need not prove their case at the pleading stage; indeed, it may not be until discovery that a plaintiff has access to the evidence it needs to detail the defendant's specific intent to infringe.

As a preliminary matter, because BSD fails to plead that AWS or Twitch knew of the patent, these defendants could not plausibly have intended to infringe the patent, and these claims are DISMISSED with leave to amend.

BSD's allegations, on the other hand, are sufficient to show that Amazon.com intended to infringe the '473 Patent. First, Amazon.com does not contest that it knew of the patent. Making all plausible inferences in BSD's favor from the well-pleaded facts, Amazon.com knew of the patent at least by December 2015, when the rejection of its '254 Application listed the BSD's '473 Patent as the primary prior art reference. Compl. ¶¶ 40-46.

Second, BSD's complaint pleads specific intent because the allegations go beyond mere knowledge of the patent to plausibly plead that Amazon.com knew of the infringement. *See Dali Wireless*, 2022 WL 16701926, at *6; *Bayer Healthcare*, 989 F.3d at 988. BSD alleges that despite knowledge of the '473 Patent, Amazon.com repeatedly tried to convince the USPTO that its technology was different (and may have been successful for at least some claims), and continued to use the allegedly infringing technology for the past eight years. Combined with the well-pleaded allegations and claim charts that assert Amazon.com's technology infringes, it is plausible to infer that Amazon.com knew that its technology infringed the '473 Patent yet willfully continued its infringing behavior. While the allegations are not nearly as strong as the jury findings affirmed by the Federal Circuit in *Arctic Cat Inc. v. Bombardier Recreational Products Inc.*, 876 F.3d 1350, 1371 (Fed. Cir. 2017), where the evidence showed the defendant knew of the patents, "conducted only a cursory analysis of the patents, waited years before seeking advice of . . . counsel, and unsuccessfully tried to buy the asserted patents," the *Artic Cat* opinion arose after a jury trial, while the pleadings here are pre-discovery and are bound to be less detailed.

United States District Court
Northern District of California

The facts here are also different from *Fortinet*, 543 F. Supp. 3d at 841-42, where the Honorable Edward M. Chen found that the plaintiff failed to plausibly plead intent[2] by alleging that the defendant was unwilling to and repeatedly refused to discuss licensing the plaintiff's patents. Here, of course, the allegations assert that Amazon.com has known of the patent for eight years, has repeatedly tried to convince the USPTO that its own technology was different, and has used the allegedly infringing technology since that point. And this case also differs from *Finjan, Inc. v. Cisco Systems Inc.*, No. 17-CV-00072-BLF, 2017 WL 2462423, at *5 (N.D. Cal. June 7, 2017), where the Honorable Beth Labson Freeman found thathere were "no specific factual allegations about [the defendant's] specific intent" aside from "conclusory allegations of knowledge and infringement," because here the allegations are specific and nonconclusory.

The defendants' sole argument to the contrary is unconvincing. They assert that BSD did not send a pre-suit letter to the defendants and so cannot show specific intent to infringe. But whether a pre-suit letter alerted a defendant to infringement goes to knowledge, not to specific intent, and the defendants do not contest that Amazon.com knew of the patent. Additionally, the reasoning is flawed even as to knowledge: there are "circumstances where willful infringement is properly alleged despite the absence of a notice letter" such as where "the alleged infringer . . . learned of the patent in a previous lawsuit [] [o]r . . . had a prior license." *Sonos*, 591 F. Supp. 3d at 644; *see also Juniper Networks*, 562 F. Supp. 3d at 381 (finding pre-suit knowledge where the plaintiff alleged the patent was the subject of previous litigation in the "tight-knit industry" and that the defendant was connected to a party that disclosed the patent's existence). This is one of those circumstances, because Amazon.com learned of the suit through prior prosecution.

Finally, in its opposition BSD relies on two cases that it says shows that a plaintiff can plead induced infringement by plausibly alleging that the defendant knew of the patent and that

[2] This case more specifically addressed whether the plaintiff pleaded *egregious* intent, which as discussed in the following section, *infra* Part III, is not necessary at this stage. At least one court in this district has specifically noted that at the pleading stage, pleading egregiousness is the same as pleading specific intent. *See Core Optical Techs., LLC v. Juniper Networks Inc.*, 562 F. Supp. 3d 376, 381 (N.D. Cal. 2021) (hereinafter "*Juniper Networks*") (subsequent history omitted) ("As for egregiousness, a patentee need only plead that the defendant engaged in 'deliberate or intentional infringement.'" (quoting *Eko Brands*, 946 F.3d at 1378)).

the claims infringe. *See Lifetime Indus., Inc. v. Trim-Lok, Inc.*, 869 F.3d 1372, 1379–80 (Fed. Cir. 2017); *Lytone Enter., Inc. v. Agrofresh Sols., Inc.*, No. 1:20-cv-678, 2021 WL 534868, at *5 (D. Del. Feb. 12, 2021). These cases concerned the standard for induced infringement, not willful infringement, and so I do not rely on them in making my finding. *See, e.g.*, *SRI Int'l*, 14 F.4th at 1329 ("To be clear, a finding of induced infringement does not compel a finding of willfulness. Indeed, the standard required for willful infringement is different [from] that required for induced infringement."). However, their reasoning does makes sense in the context of willful infringement. Where a plaintiff plausibly alleges that the defendant knew of the patent—and, as here, knew of the patent for years—and pleads that the technology infringed in many clear ways, it is reasonable to infer that the defendant knew that it was infringing the patent. At this early stage in litigation, a plaintiff is not expected to plead in detail all the facts that may support intent; that is the purpose of discovery. For now, it is sufficient to plausibly allege that Amazon.com knew of the patent, that it tried several times to convince the USPTO that the technology did not infringe, that the technology did in fact infringe, and that it has infringed for eight years despite its knowledge. Accordingly, BSD plausibly alleges willful infringement.[3]

For those reasons, then, BSD's allegations that Amazon.com willfully infringed the '473 Patent are plausible. The motion is DENIED as to this argument.

III. EGREGIOUSNESS

The parties disagree over whether plaintiffs must plead egregious conduct in willful infringement claims, and if required, whether BSD sufficiently pleaded egregiousness here.

A plaintiff alleging patent infringement may recover enhanced damages, but her ability to do so relies on the existence and extent of egregious behavior by the defendant. *See Sonos*, 591 F. Supp. 3d at 643–45. Because the Federal Circuit has not provided clear post-*Halo* guidance on

[3] In opposition, BSD argues that it pleaded specific intent by alleging that Amazon.com was willfully blind to the patent. But it is not clear why allegations of willful blindness would show specific intent, as willful blindness is a theory of knowledge. *See Glob.-Tech Appliances, Inc. v. SEB S.A.*, 563 U.S. 754, 770–71 (2011). Notably *Fortinet*, 543 F. Supp. 3d at 841, is not to the contrary: in stating that willful blindness "may suffice to state a claim for willful infringement," Judge Chen cited *TC Tech. LLC v. Sprint Corp.*, No. 16-CV-153-RGA, 2019 WL 529678, at *4 (D. Del. Feb. 11, 2019), which cited *Intel Corp. v. Future Link Sys., LLC*, 268 F. Supp. 3d 605, 623 (D. Del. 2017), where willful blindness went to the knowledge of the defendant.

United States District Court
Northern District of California

pleading requirements for willful infringement claims, courts in this district are somewhat divided over whether plaintiffs must plead egregiousness for a willful infringement claim to survive a motion to dismiss. *See id.* at 644 n.1 (collecting cases).

In *Sonos*, the Honorable William Alsup relied on the Federal Circuit's holdings in *Eko* and *Halo* that enhanced damages should not be addressed until "after infringement is *proven*" to reason that so long as a willful infringement plaintiff adequately pleads willfulness, the plaintiff "need *not* go further and specify the further aggravating circumstances warranting enhanced damages." *Id.* at 644-45; *see also Eko Brands*, 946 F.3d at 1378 ("The question of enhanced damages is addressed by the court once an affirmative finding of willfulness has been made." (citing *Halo*, 579 U.S. at 105-06)); *Juniper Networks*, 562 F. Supp. 3d at 381 ("[T]he ultimate question whether the alleged conduct is sufficiently egregious to warrant enhanced damages is for the Court to decide at summary judgment or after trial."). I am persuaded by this reasoning; as stated in *Sonos*, "[t]he full extent of egregious behavior is [often] unknown at the pleading stage." *Sonos*, 591 F. Supp. 3d at 645. Accordingly, BSD was not required to plead specific acts of egregiousness to survive a motion to dismiss; its detailed allegations of specific intent are sufficient at this stage.

Accordingly, the motion is DENIED as to this argument.

CONCLUSION

The motion to dismiss is GRANTED in part and DENIED in part. BSD may file any amended complaint within 20 days of the issuance of this order.

IT IS SO ORDERED.

Dated: July 27, 2023

William H. Orrick
United States District Judge

EXHIBIT G

US 20180357282A1

(19) United States
(12) Patent Application Publication
AMBARTSUMOV et al.

(10) Pub. No.: US 2018/0357282 A1
(43) Pub. Date: Dec. 13, 2018

(54) SYSTEM AND METHOD FOR EFFICIENTLY HANDLING QUERIES

(71) Applicant: **KMS Lighthouse Ltd.**, Bnei Braq (IL)

(72) Inventors: **Stanislav AMBARTSUMOV**, Irkutsk (RU); **Lev GARKAVY**, Yavne (IL); **Moshe RECANATI**, Givat Shmuel (IL)

(73) Assignee: **KMS Lighthouse Ltd.**, Bnei Braq (IL)

(21) Appl. No.: **15/620,152**

(22) Filed: **Jun. 12, 2017**

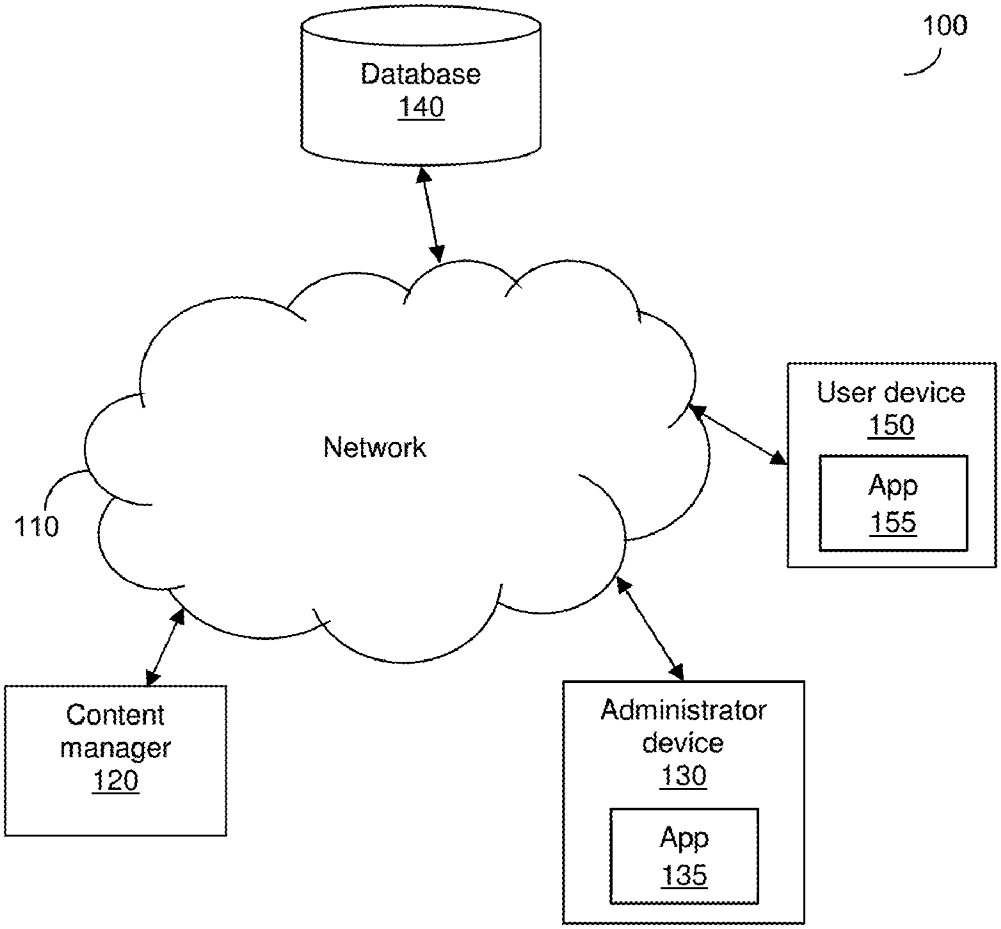
Publication Classification

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. Cl.**
CPC *G06F 17/30474* (2013.01)

(57) ABSTRACT

A system and method for handling queries. The method includes receiving a query; determining, based on the received query, a best matching template question; selecting, from among a plurality of question templates, a question template for responding to the query, wherein the selected question template includes the determined best matching template question and a set of answers assigned to the selected question template; and generating, based on the selected question template, a response to the query.



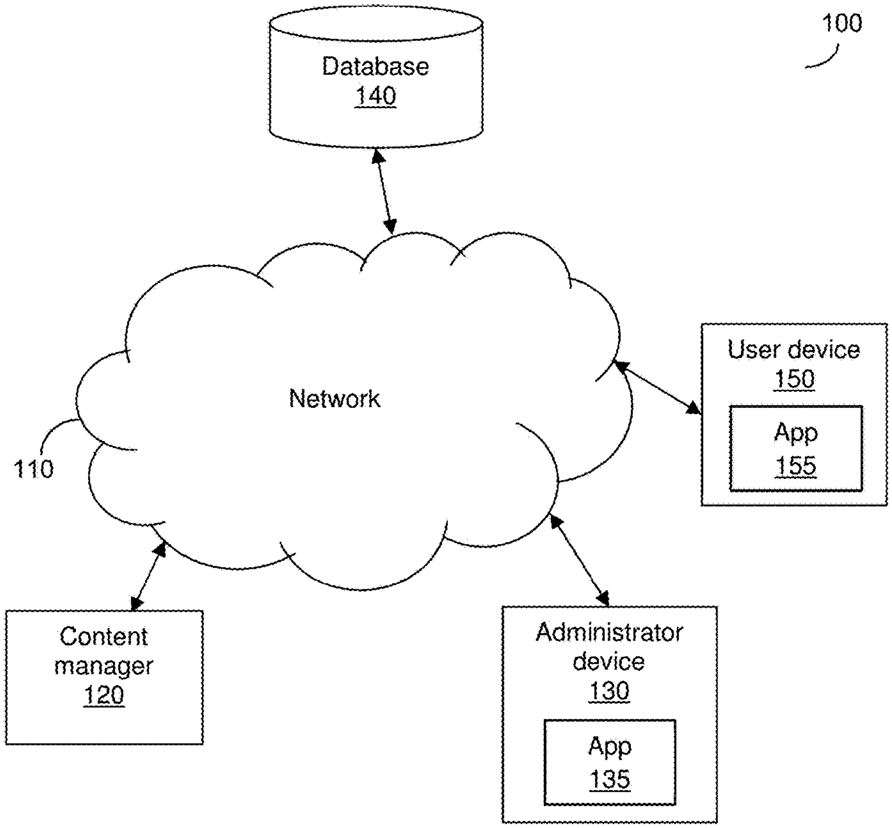

Database
140
100
Network
110
User device
150
App
155
Content
manager
120
Administrator
device
130
App
135

FIG. 1

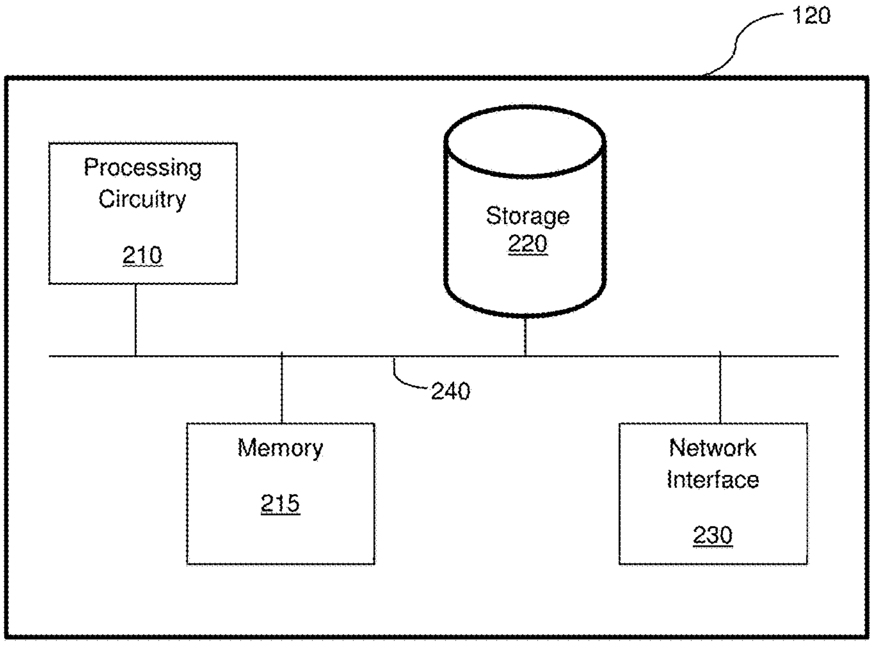

120
Processing Circuitry
210
Storage
220
240
Memory
215
Network Interface
230

FIG. 2

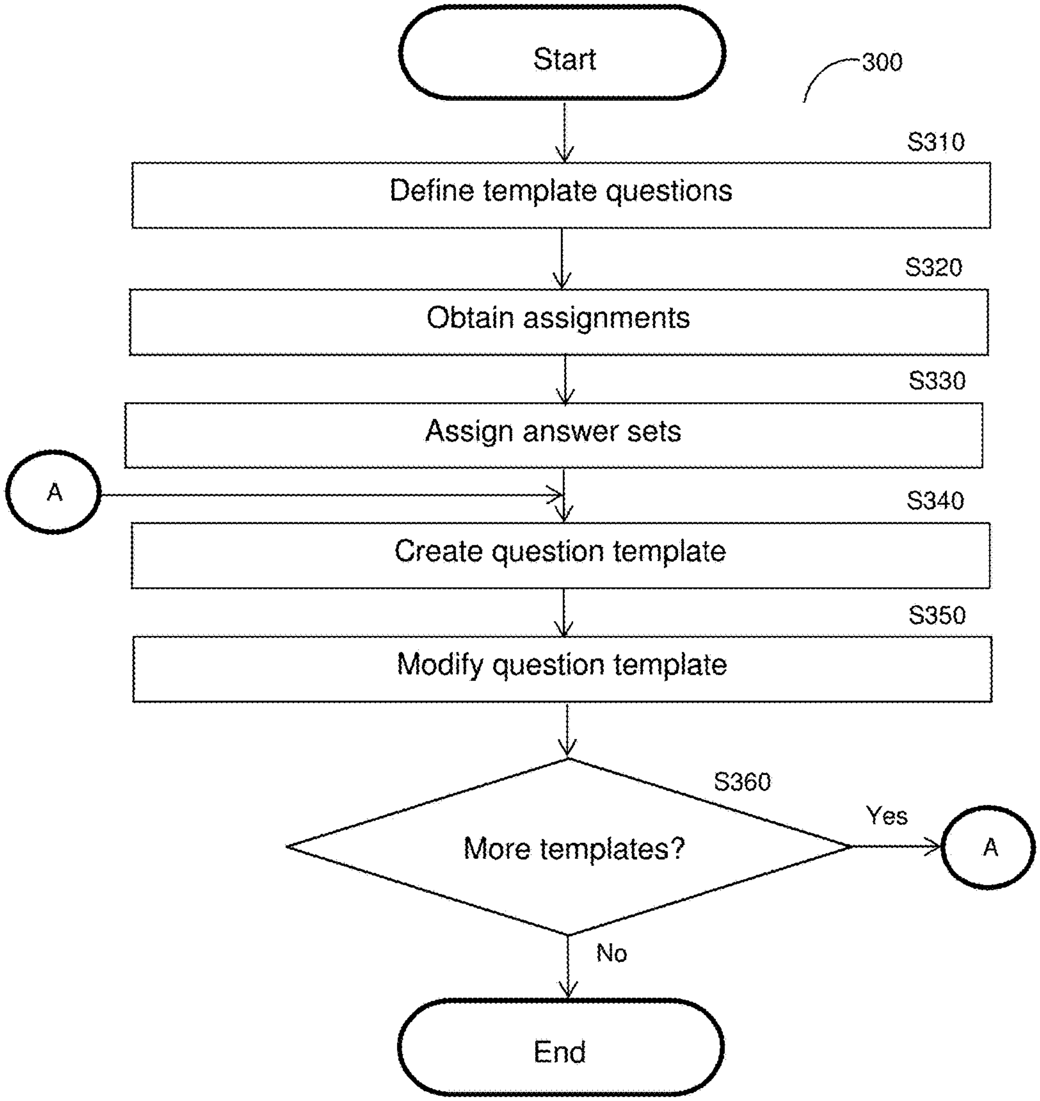

Start
300
S310
Define template questions
S320
Obtain assignments
S330
Assign answer sets
A
S340
Create question template
S350
Modify question template
S360
More templates?
Yes
A
No
End

FIG. 3

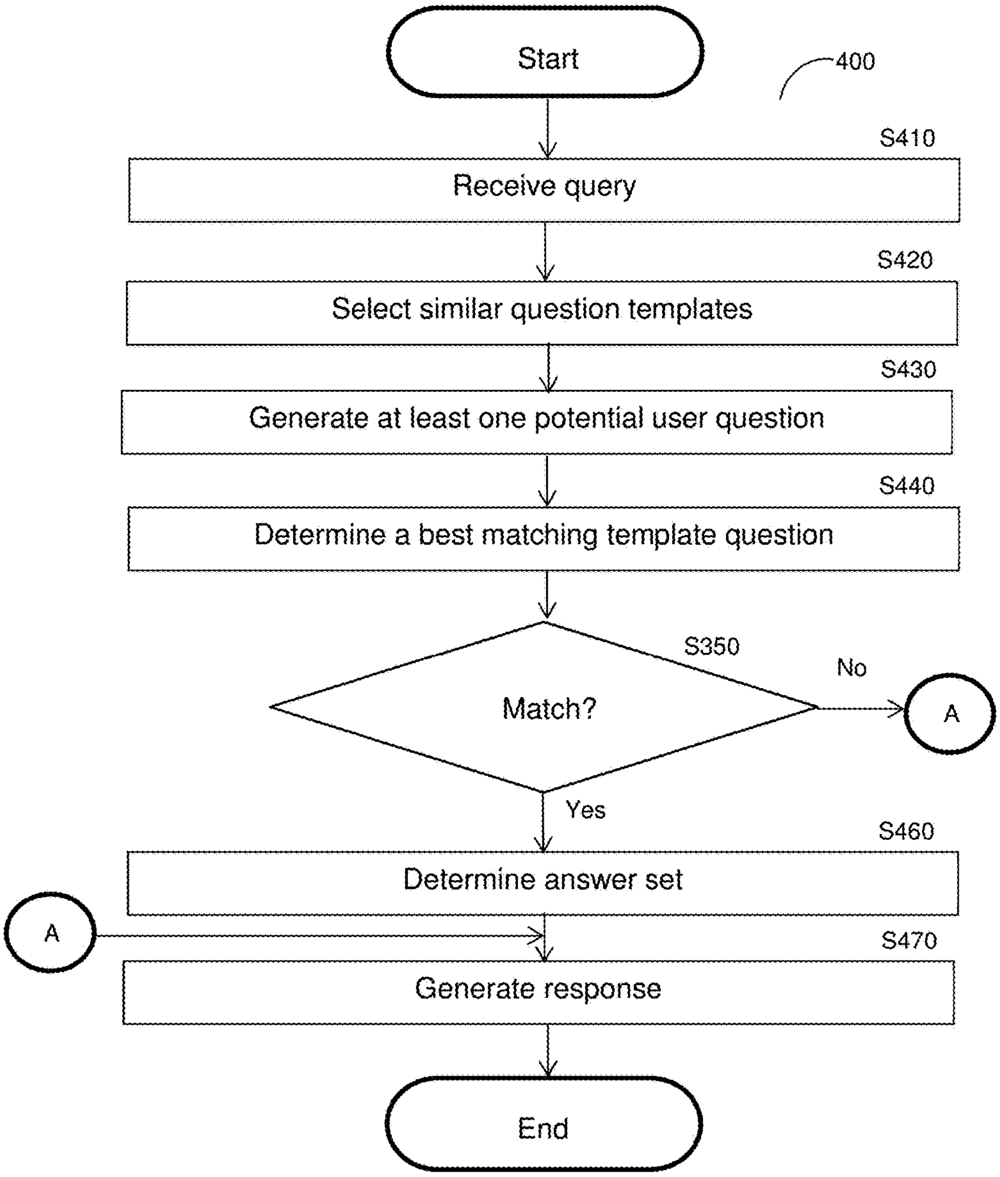
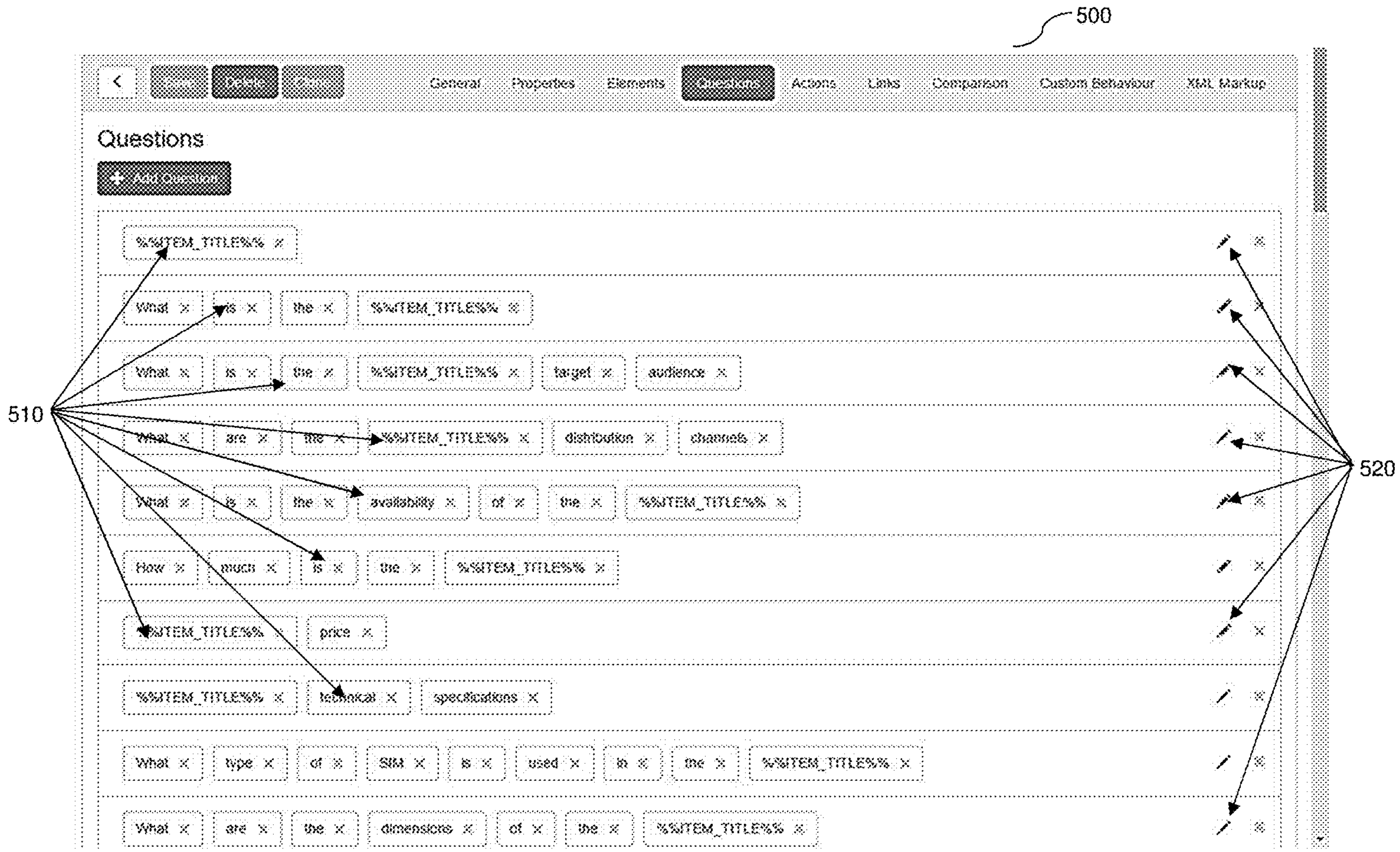

Start
400
S410
Receive query
S420
Select similar question templates
S430
Generate at least one potential user question
S440
Determine a best matching template question
S350
Match?
No
A
Yes
S460
Determine answer set
A
S470
Generate response
End

FIG. 4



%%ITEM_TITLE%%

What is the %%ITEM_TITLE%%

What is the %%ITEM_TITLE%% target audience

What are the %%ITEM_TITLE%% distribution channels

What is the availability of the %%ITEM_TITLE%%

How much is the %%ITEM_TITLE%%

%%ITEM_TITLE%% price

%%ITEM_TITLE%% technical specifications

What type of SIM is used in the %%ITEM_TITLE%%

What are the dimensions of the %%ITEM_TITLE%%

FIG. 5

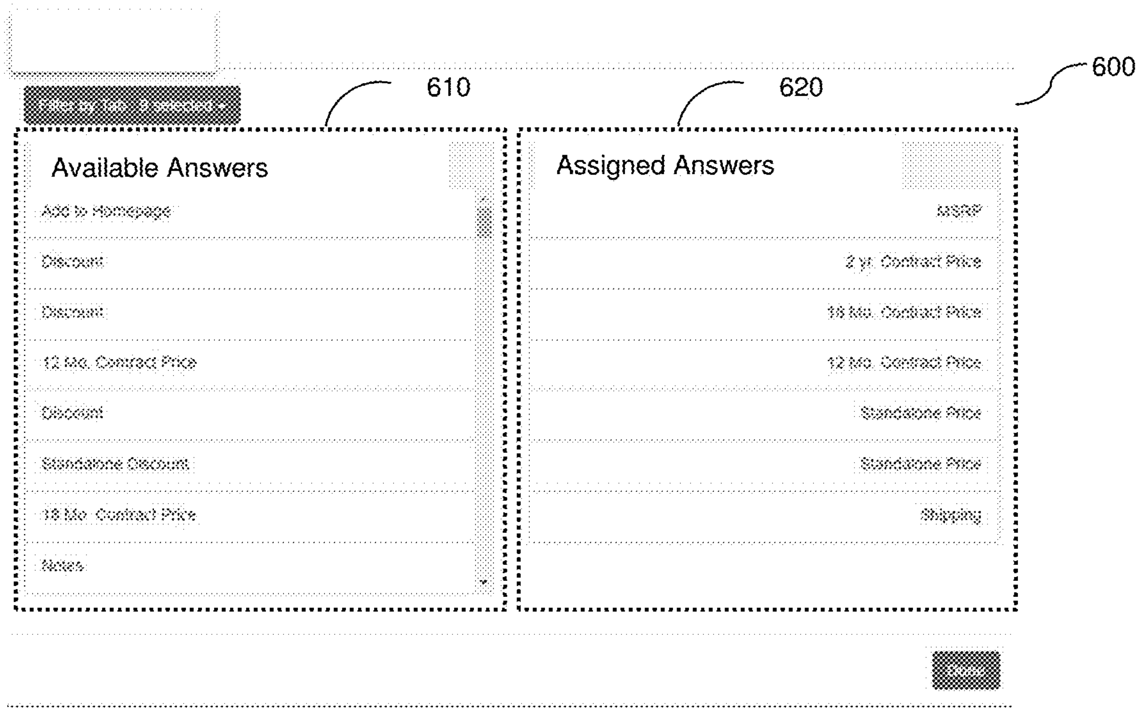
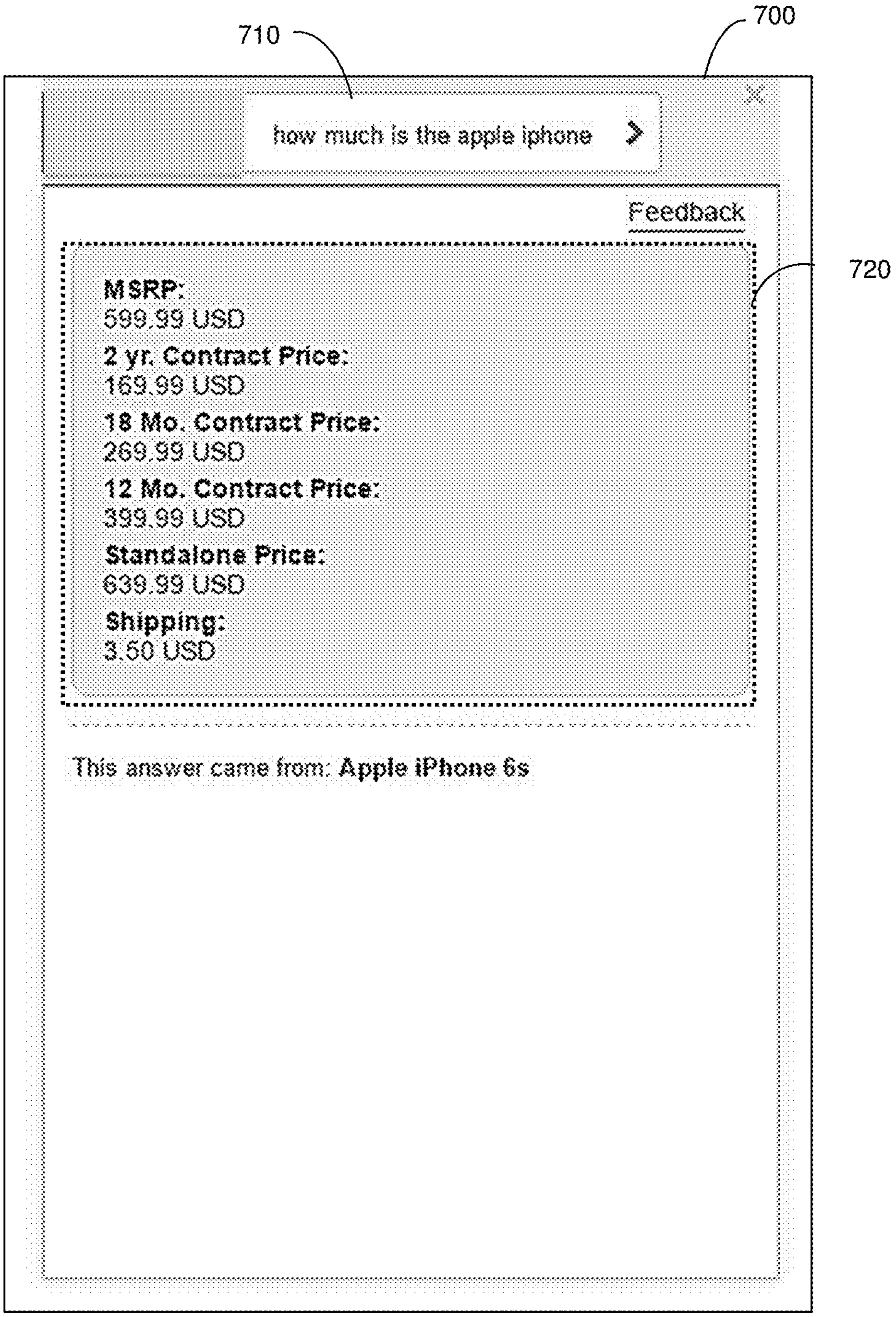

600
610
620
Available Answers
Add to Homepage
Discount
Discount
12 Mo. Contract Price
Discount
Standalone Discount
18 Mo. Contract Price
Notes
Assigned Answers
MSRP
2 yr. Contract Price
18 Mo. Contract Price
12 Mo. Contract Price
Standalone Price
Standalone Price
Shipping

FIG. 6

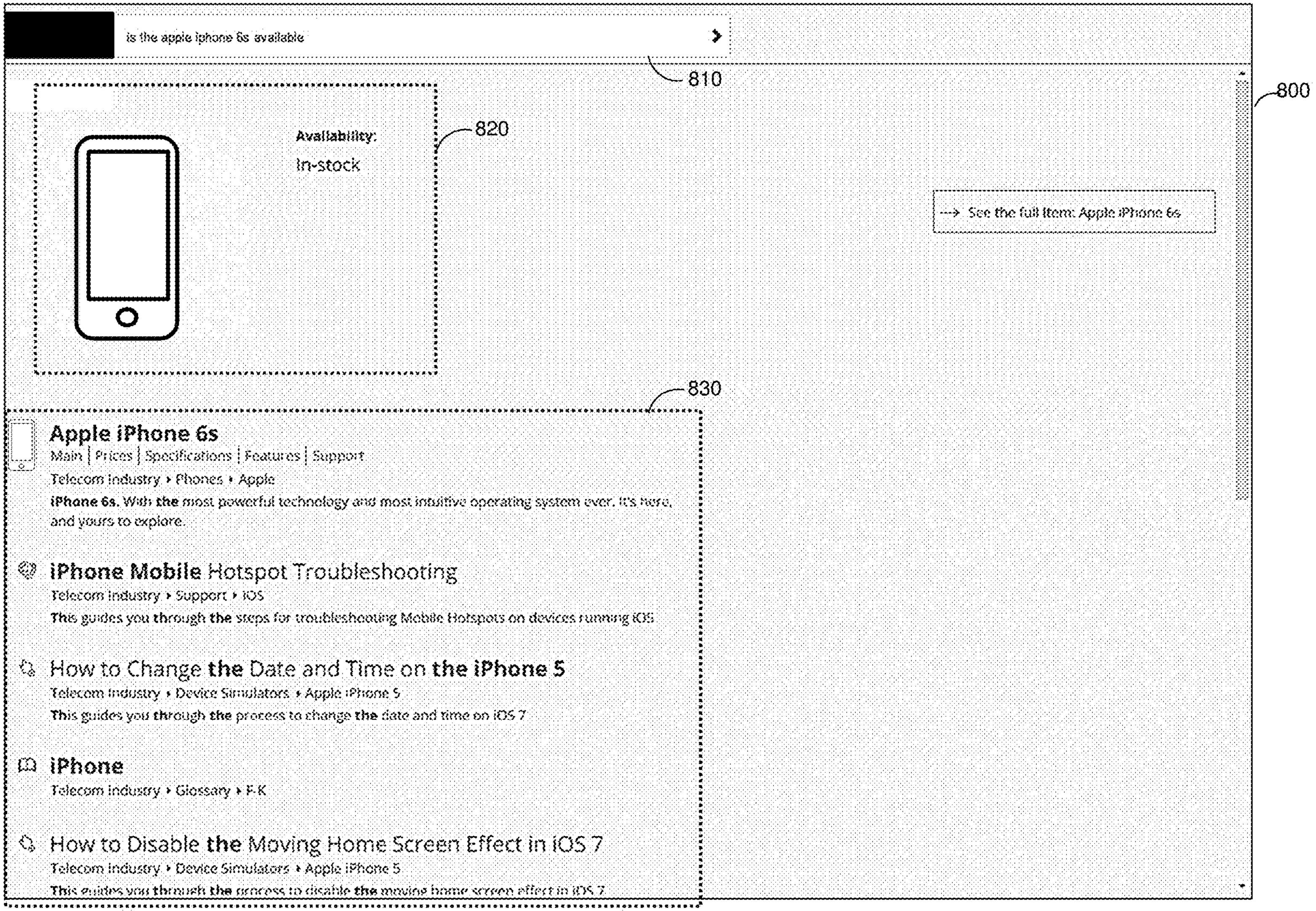


how much is the apple iphone >

MSRP:
599.99 USD
2 yr. Contract Price:
169.99 USD
18 Mo. Contract Price:
269.99 USD
12 Mo. Contract Price:
399.99 USD
Standalone Price:
639.99 USD
Shipping:
3.50 USD

This answer came from: **Apple iPhone 6s**

FIG. 7



is the apple iphone 6s available

Availability:
In-stock

→ See the full item: Apple iPhone 6s

Apple iPhone 6s
Main | Prices | Specifications | Features | Support
Telecom Industry › Phones › Apple
iPhone 6s. With **the** most powerful technology and most intuitive operating system ever. It's here, and yours to explore.

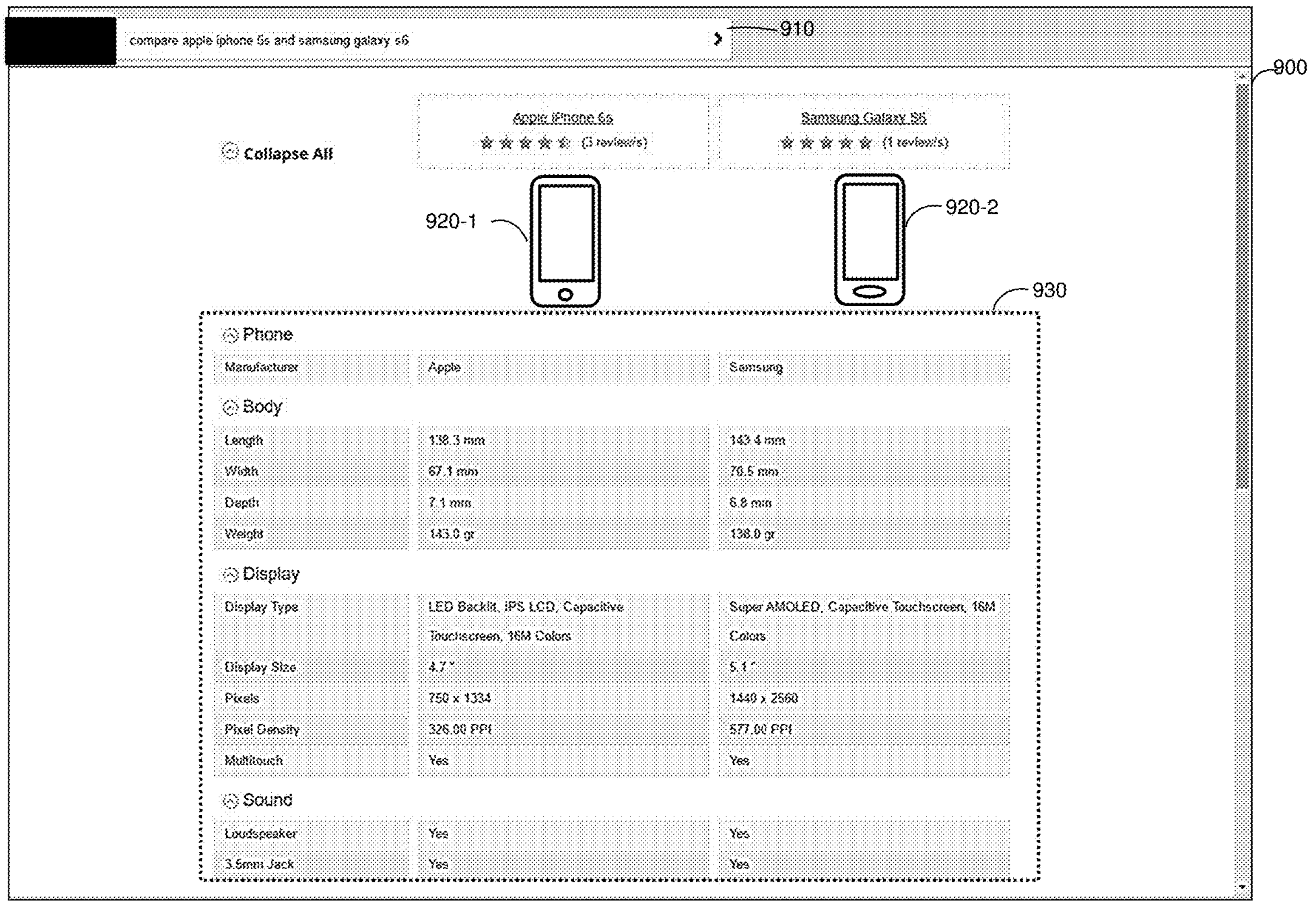
iPhone Mobile Hotspot Troubleshooting
Telecom Industry › Support › iOS
This guides you through **the** steps for troubleshooting Mobile Hotspots on devices running iOS

How to Change **the** Date and Time on **the iPhone 5**
Telecom Industry › Device Simulators › Apple iPhone 5
This guides you through **the** process to change **the** date and time on iOS 7

iPhone
Telecom Industry › Glossary › F-K

How to Disable **the** Moving Home Screen Effect in iOS 7
Telecom Industry › Device Simulators › Apple iPhone 5
This guides you through **the** process to disable **the** moving home screen effect in iOS 7

FIG. 8



compare apple iphone 6s and samsung galaxy s6

	Apple iPhone 6s ★★★★★ (3 review/s)	Samsung Galaxy S6 ★★★★★ (1 review/s)
Phone		
Manufacturer	Apple	Samsung
Body		
Length	138.3 mm	143.4 mm
Width	67.1 mm	70.5 mm
Depth	7.1 mm	6.8 mm
Weight	143.0 gr	138.0 gr
Display		
Display Type	LED Backlit, IPS LCD, Capacitive Touchscreen, 16M Colors	Super AMOLED, Capacitive Touchscreen, 16M Colors
Display Size	4.7 "	5.1 "
Pixels	750 x 1334	1440 x 2560
Pixel Density	326.00 PPI	577.00 PPI
Multitouch	Yes	Yes
Sound		
Loudspeaker	Yes	Yes
3.5mm Jack	Yes	Yes

FIG. 9

SYSTEM AND METHOD FOR EFFICIENTLY HANDLING QUERIES

TECHNICAL FIELD

[0001] The present disclosure relates generally to providing responses to queries, and more particularly to efficient retrieval of responses to free text queries.

BACKGROUND

[0002] Customer Search Representatives (CSRs) traditionally function by verbally answering customers' questions or concerns over the phone, which typically provides the customer with the assistance needed while providing the customer with human interaction. However, responding to queries via CSRs may be costly, inefficient, and require a live human operator at all times. Additionally, responses provided by CSRs are subject to human error.

[0003] Some automated solutions for providing responses to queries exist. As an example, some solutions allow users to select from a predetermined menu of options representing questions associated with answers stored in a database. However, such menu-based solutions may be difficult to navigate, and do not allow the user to freely enter queries (i.e., if a question is not among the menu options, such solutions typically cannot provide an appropriate answer).

[0004] As another example, search engines may be utilized to search for related content based on a query. However, existing search engine-based solutions typically only superficially recognize content of the query by searching for content featuring words matching words included in the query. Results obtained via such solutions are often only tangentially related (or unrelated entirely) to the information sought by the user.

[0005] As yet another example, some existing solutions allow for receiving textual queries and providing closely related responses by searching through a comprehensive knowledge database. However, such comprehensive knowledge databases must typically include a large number of specific queries along with associated answers, resulting in significant use of storage and inefficiencies in retrieving responses.

[0006] It would therefore be advantageous to provide a solution that would overcome the challenges noted above.

SUMMARY

[0007] A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the term "some embodiments" or "certain embodiments" may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

[0008] Certain embodiments disclosed herein include a method for handling queries. The method comprises: receiving a query; determining, based on the received query, a best matching template question; selecting, from among a plurality of question templates, a question template for responding to the query, wherein the selected question template includes the determined best matching template question and a set of answers assigned to the selected question template; and generating, based on the selected question template, a response to the query.

[0009] Certain embodiments disclosed herein also include a non-transitory computer readable medium having stored thereon causing a processing circuitry to execute a process, the process comprising: receiving a query; determining, based on the received query, a best matching template question; selecting, from among a plurality of question templates, a question template for responding to the query, wherein the selected question template includes the determined best matching template question and a set of answers assigned to the selected question template; and generating, based on the selected question template, a response to the query.

[0010] Certain embodiments disclosed herein also include a system for handling queries. The system comprises: a processing circuitry; and a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: receive a query; determine, based on the received query, a best matching template question; select, from among a plurality of question templates, a question template for responding to the query, wherein the selected question template includes the determined best matching template question and a set of answers assigned to the selected question template; and generate, based on the selected question template, a response to the query.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

[0012] FIG. **1** is a network diagram utilized to describe the various disclosed embodiments.

[0013] FIG. **2** is a schematic diagram of a content manager according to an embodiment.

[0014] FIG. **3** is a flowchart illustrating a method for creating question templates according to an embodiment.

[0015] FIG. **4** is a flowchart illustrating a method for providing responses to queries using question templates according to an embodiment.

[0016] FIG. **5** is a screenshot illustrating a graphical user interface for providing question-related inputs.

[0017] FIG. **6** is a screenshot illustrating a graphical user interface for providing answer-related inputs.

[0018] FIG. **7** is a screenshot illustrating a response to a user query based on a question template.

[0019] FIG. **8** is a screenshot illustrating a response to a user query.

[0020] FIG. **9** is a screenshot illustrating a comparison chart provided as a response to a user query.

DETAILED DESCRIPTION

[0021] It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

[0022] The various disclosed embodiments include methods and systems for efficiently handling queries. In an example implementation, a customer service department can predefine question templates for generating questions that allow for efficient responses based on free text or other unstructured queries from customer service representatives or end users, typically within a few seconds. The question templates may further include answers used for providing the responses.

[0023] In an embodiment, template inputs are received. The template inputs may include inputs defining template questions, inputs defining answers, and inputs assigning question templates to subjects. In an optional embodiment, the template questions, the answers, or both, may be modified based on at least one modification rule. Based on the received template inputs, at least one question template is created.

[0024] In an embodiment, when the question templates are created, a query may be received. A user question is determined based on the received query and the question templates. A response is provided based on the determined question. The response includes at least the answer assigned to a question template including the determined question.

[0025] FIG. **1** shows an example network diagram **100** utilized to describe the various disclosed embodiments. In the example network diagram **100**, a content manager **120**, an administrative device **130**, a database **140**, and a user device **150** are communicatively connected via a network **110**. The network **110** may be, but is not limited to, a wireless, cellular or wired network, a local area network (LAN), a wide area network (WAN), a metro area network (MAN), the Internet, the worldwide web (WWW), similar networks, and any combination thereof.

[0026] The administrator device **130** is utilized by an administrator to provide inputs utilized for creating question templates. The administrator device **130** may be, but is not limited to, a personal computer, a laptop, a tablet computer, a smartphone, a wearable computing device, or any other device capable of receiving and displaying notifications. The administrator device **130** may have installed thereon an agent **135** which is configured to receive inputs from an administrator using the administrator device **130** and to send template parameters defining question templates to the content manager **120** based on the received inputs. The agent **135** may be, but is not limited to, an application such as a mobile application, a virtual application, a web application, a native application, and the like. The administrator device **130** may include a display (not shown) for displaying graphical user interfaces for receiving selections of template parameters (e.g., as shown in the example screenshots of FIGS. **4** and **5**).

[0027] The database **140** stores at least question templates as well as assignments of question templates. Each question template includes, but is not limited to, at least one template question, where each template question is defined by a set of template question parameters. Each question template is assigned to at least one subject and has a set of at least one answer assigned to it. Each answer may include information, visual content, or both.

[0028] The user device **150** may be, but is not limited to, a personal computer, a laptop, a tablet computer, a smartphone, a wearable computing device, or any other device capable of receiving and displaying notifications. The user device **150** may be, for example, a user device of a CSR or a customer seeking an answer to a question. The user device **150** have installed thereon an agent **155** which is configured to receive user inputs indicating queries, to send queries based on the user inputs, and to receive responses to the queries from the content manager **120**. The agent **155** may be, but is not limited to, an application such as a mobile application, a virtual application, a web application, a native application, and the like. In an example implementation, the agent **155** may be a web browser.

[0029] In an embodiment, the content manager **120** is configured to receive, from the administrator device **130**, template inputs. The template inputs may include, but are not limited to, question-related inputs, assignment inputs, answer-related inputs, combinations thereof, a portion thereof, and the like. In a further embodiment, the content manager **120** is configured to create at least one question template based on the template inputs. Each question template includes at least one template question, is assigned to at least one subject, and has an answer set including at least one answer assigned thereto. In some embodiments, each question template may include the assigned answer set. A subject is a person, place, or thing. Example subjects include, but are not limited to, a product, a service, a company, and the like.

[0030] The question-related inputs include at least one set of template question parameters that may include, but are not limited to, subject variables, characteristic variables, terms forming a structure of the question (e.g., “who”, “what”, “when”, “where”, “why”, “how”, “is”, “are”, “the”, “does”, “can”, etc.), terms indicating degree (e.g., “much”, “many”, “amount”, “number”, etc.), images or other multimedia content submitted as a query, and any other information utilized to define questions. Each set of template question parameters defines a template question.

[0031] In some implementations, a set of template question parameters may define a unique template question that is unique to a particular subject. Such a unique template question typically does not include variable template question parameters (e.g., subject variables or characteristic variables), and is typically not applied to multiple subjects.

[0032] Each subject variable represents a subject of a template question, and is replaced with the subject when utilized for determining user questions, thereby allowing for use of the same template question to define different potential user questions for different subjects. Each characteristic variable represents a characteristic or component of the subject, and is replaced with the characteristic or component when utilized for determining user questions, thereby allowing for use of the same set of template question parameters to define different potential user questions for different characteristics or components of a subject.

[0033] As a non-limiting example, a template question may include the set of template question parameters “What is the availability of the [Subject_Title]”, where “[Subject_Title]” is the subject variable, such that the user questions determined based on the set of template question parameters may include "What is the availability of the iPhone 7" and "What is the availability of the "Samsung Galaxy S8" when the question is assigned to the subjects "iPhone 7" and "Samsung Galaxy S8," respectively.

[0034] As another non-limiting example, a template question may include the template question parameters "Does [Subject_Title] have [Characteristic_Title]", where "[Subject_Title]" is the subject variable and "[Characteristic_Title]" is the characteristic variable, such that the user questions determined based on the set of template question parameters may include "Does iPhone 6 have camera", "Does iPhone 6 have headphone jack", "Does Samsung Galaxy S8 have camera", and "Does Samsung Galaxy S8 have headphone jack" when the template question is applied to combinations of one of the subjects "iPhone 6" or "Samsung Galaxy S8" with one of the characteristics "camera" or "headphone jack".

[0035] The assignment inputs indicate, for each question template to be created, at least one subject to which the question template is assigned. The assignment inputs may further indicate a characteristic to which one or more of the template questions are assigned. The characteristics to which a template question is assigned are characteristics for which the template question (and, consequently, corresponding answers) may be relevant. As a non-limiting example, for a characteristic of "business hours", template questions such as "What are the [Characteristic_Title] for [Subject_Title]" and "When is [Subject_Title] open" may be assigned to the characteristic such that the "[Characteristic_Title]" is replaced with "business hours" when determining user questions and the [Subject_Title] may be replaced with subjects such as "Dave's Hardware" (i.e., a particular store) or "West Orange NJ ABC Chain Restaurant" (i.e., a branch of a company having multiple stores).

[0036] The answer-related inputs include or indicate (e.g., by indicating a selection or a location in storage) a set of at least one answer and an assignment of each answer set to at least one question template as well as to at least one subject. Each answer includes content which may further include, but is not limited to, images, videos, audio, information, graphs, charts, combinations thereof, and the like. As a non-limiting example, the answer-related inputs may include information related to smartphones such as price, subscription costs, presence or absence of certain components, an image of the phone, a video of the phone in use, a combination thereof, or any other content that may be utilized to answer a user's question. Example answer content is illustrated in FIGS. **6**, **7**, and **8**, each described further herein below. The answer-related inputs may further indicate assignments of such information to appropriate question templates (e.g., price information for the iPhone 7 may be assigned to the combination of a question template defining price-related questions and the subject "iPhone 7").

[0037] Defining the answer for each question template including multiple template questions allows for returning the same answer for substantively similar questions. As a non-limiting example, an answer including the text "MSRP: $700" may be returned for both the questions "What is the cost of iPhone 6" and "How expensive is iPhone 6" that are defined in a question template assigned to the subject "iPhone 6".

[0038] In an embodiment, the content manager **120** may be configured to store the created question templates as well as assignment data indicating assignments of the created question templates in, e.g., the database **140**. It should be noted that, in an embodiment, only template questions (i.e., as opposed to each specific instance of a template question) are stored in the database **140** and utilized to determined potential user questions as needed, thereby resulting in lower usage of computing resources related to storing variations of questions. As a non-limiting example, the template question "what is the price of [Subject_Title]" may be stored (and, optionally, one example variant per subject such as "what is the price of Nintendo Switch"), but other variants such as "what is the cost of [Subject_Title]", "what price [Subject_Title]", and "what is [Subject_Title] price" do not need to be stored to match queries to the template question.

[0039] In an embodiment, the content manager **120** may be configured to collect data related to changes to a question template by the administrator of the administrator device **130**. The change data may include, but is not limited to, times of changes, portions of the question template modified (e.g., questions, answers, assignments, etc.), and the like. In a further embodiment, the content manager **120** may be configured to generate and send a notification prompting the administrator to revise the questions of the question template when a number of changes to questions in the question template is above a predetermined threshold during a given time period (i.e., when changes are constantly or otherwise frequently occurring, which may indicate that the questions of the question template require closer attention).

[0040] In an embodiment, the content manager **120** may be configured to automatically modify created question templates. The modifications may include, but are not limited to, adding questions, changing existing questions, deleting questions (e.g., deleting duplicate questions), adding answers, assigning questions to additional subjects, removing assignments of questions to subjects, and the like. Changing the existing questions may include, but is not limited to, correcting spelling and grammar of template question parameters. In a further embodiment, the content manager **120** may be configured to send a notification to the administrator device **130** prompting the administrator using the administrator device **130** to confirm the modification. If the modification is confirmed, the modification may be performed automatically.

[0041] In an embodiment, the content manager **120** may be configured to delete duplicate questions. Questions may be duplicates if they meet matching requirements such as, e.g., each template parameter of a first template question being the same as or a synonym of a corresponding template parameter of a second template question (for example, as indicated in a list of terms and corresponding synonyms), each template parameter of a first template question matching a corresponding template parameter of a second template question except for a misspelling or truncation (for example, as determined based on at least one of a stemmer, a phonetic analyzer, a morphology analyzer, a spellchecker, etc.), differences between two template questions not being substantively different (e.g., the presence of punctuation such as "?", capitalization, words having at most a predetermined number of characters, or stop words such as "is", "of", and "and", occurring in one of the template questions but not in the other), a combination thereof, and the like. Template questions may match if the template parameters are in a different order so long as the above-noted matching requirements are met. As a non-limiting example, the questions "Who is the president of the United States of America?" and "who is USA's President" may be determined to match despite having a different order as well as differences in capitalization, synonyms (i.e., "United States of America" and "USA's"), use of stop words such as "the", and use of "?" as punctuation.

[0042] In an embodiment, the content manager **120** may be configured to suggest sets of template question parameters for question templates. The suggested sets of template question parameters may be based on, but not limited to, popular questions that are not defined in the question template (e.g., questions that have been asked above a threshold number of times); template questions defined in other question templates related to the same subject, characteristic, or both; a combination thereof (e.g., popular questions related to the same subject, characteristic, or both); and the like.

[0043] In an embodiment, when question templates have been created and assigned, the content manager **120** is configured to receive, from the user device **150**, a query, and to determine, based on the received query and the created question templates, a question indicated by the query. The question indicated by the query may be determined with respect to one or more of the sets of template question parameters of the created question templates as well as the characteristic and subjects to which each question template is assigned.

[0044] In a further embodiment, the content manager **120** is configured to determine the question indicated by the query by determining a template question matching the query. In yet a further embodiment, the content manager **120** may be configured to identify a subject indicated in the query, a characteristic indicated in the query, or both, and to compare the query only to template questions defined in question templates assigned to the identified subject, template questions assigned to the identified characteristic, or a combination thereof.

[0045] In a further embodiment, identifying a subject or characteristic in the query may include applying one or more semantic rules to a potential subject or characteristic in the query. A potential subject or characteristic may be related to a predetermined subject or characteristic, but is not the same or a synonym. As a non-limiting example, a semantic rule for the potential subject "iphone" (i.e., an indicator of a smart phone but without a specific version) may result in identifying the subject based on release date such that the most recent version of the iPhone (e.g., the subject "iPhone 7") is identified.

[0046] Comparing the query to a template question may include, but is not limited to, replacing the subject, the characteristic, or both, identified in the query, with the respective identified subject variable, characteristic variable, or both, of the template question to generate a potential user question, and comparing the query to the potential user question to determine if they match.

[0047] A query and a potential user question may match if they meet matching requirements such as, e.g., each term of the potential user question being the same as or a synonym of a corresponding term in the query, each term of the potential user question matching a corresponding term in the query except for a misspelling or truncation, the potential user question and the query not being substantively different (e.g., with respect to punctuation, capitalization, short words, and stop words), a combination thereof, and the like. A query and a potential user question may match if the template question parameters of the question are in a different order than that of terms in the query so long as the above-noted matching requirements are met.

[0048] In an embodiment, the content manager **120** is configured to send, to the user device **150**, a response to the query. In a further embodiment, the response includes a set of answers assigned to the question template and subject of the potential user question that matches the query. In a further embodiment, if multiple potential user questions are determined to match the query, the answers assigned to the potential user question of the highest-ranking question template may be included in the response.

[0049] Accordingly, in an embodiment, the content manager **120** is configured to provide only one set of answers to the user's query. Providing only one set of answers to a query allows for providing only the answers that likely provide the specific information the user is looking for as opposed to, e.g., a plurality of search results that the user must sift through, with some results being unrelated or only tangentially related to the user's query. The response may optionally further include search results (e.g., search results obtained by submitting the query to a search engine) as supplemental information to the answer set. Further, by only returning responses including a set of answers to a matching question, only relevant answers are provided, as opposed to simply the closest or "best" answer as determined based on, e.g., the presence of select keywords.

[0050] In an embodiment, if no potential user question matches the query, no corresponding answers are included in the response. In a further embodiment, if no template question matches the query, the response includes search results. In another embodiment, the content manager **120** may be configured to generate a notification prompting the administrator using the administrator device **130** to add the query to a question template as a template question.

[0051] In another embodiment, if no potential user question matches the query, a set of answers corresponding to a partially matching potential user question may be included in the response such that the response includes answers that are likely to provide the information sought by the user despite the failure to find a match. In a further embodiment, a potential user question partially matches the query if there is partial matching between a subject identified in the query and a subject in the potential user question, the terms in the query and in the potential user question otherwise meet the above-noted matching requirements (i.e., terms other than terms indicating the subject), and there is no other matching potential user question. As a non-limiting example, for a question template including the template question "Who is the [prime_minister] of [Subject_Title]" assigned to the subject "the State of Israel" (i.e., such that a potential user question is "who is the prime minister of the State of Israel"), a query "who is the prime minister of Israel" may be considered partially matching to the potential user question such that a set of answers assigned to the template question (e.g., a set of answers including the name of the current prime minster of Israel) is returned if there are no other matching potential user questions. However, for a query "where is the prime minister of Israel", the template question would not be partially matching (and, thus, no assigned answers would be returned) because the word "where" in the query differs from the word "who" in the template question.

[0052] It should be noted that the embodiments described herein above with respect to FIG. **1** are described with respect to one administrative device **130** merely for simplicity purposes and without limitation on the disclosed embodiments. Multiple administrative devices, multiple user devices, or both, may be equally utilized without departing from the scope of the disclosure.

[0053] Non-limiting example screenshots illustrating defining template questions, assigning a set of answers to a question template, and returning a response based on a created question template, respectively, are shown in FIGS. **5**-**7**.

[0054] FIG. **5** shows an example implementation for receiving question-related inputs. In FIG. **5**, an example screenshot **500** shows a graphical user interface utilized by an administrator to provide the question-related inputs. The example screenshot **500** shows a plurality of parameter icons **510**, each representing a template question parameter selected by an administrator. In an example implementation, the selections may be made by dragging and dropping the icons **510**. The example screenshot **500** also shows an answer assignment button **520** for each question template. The answer assignment button **520**, when clicked on, may cause display of an answer assignment interface (not shown in FIG. **5**). In the example implementation shown in FIG. **5**, subject variables are represented by the icons **510** including "%%Item_Title%%".

[0055] FIG. **6** shows an example screenshot **600** demonstrating provision of answer-related inputs. The example screenshot **600** includes a list of available answers **610** and a list of assigned answers **620**. The example screenshot **600** shows a graphical user interface, where display of the graphical user interface may be triggered in response to clicking on an answer assignment button for a question template displayed in a listing of question templates for a particular subject (e.g., as seen in FIG. **5**). In the example screenshot **600**, answers may be assigned to the question template with respect to the subject by dragging and dropping answers from the list of available answers **610** to the list of assigned answers **620**.

[0056] FIG. **7** shows an example screenshot **700** illustrating a response to a user query based on a question template. In the example screenshot **700**, a search bar **710** allows a user to input a free text query. In the example screenshot **600**, the user has entered the query "how much is the apple iphone". The search bar **710** may optionally include autocomplete options for guiding a user in choosing common or otherwise known user questions. Based on a matching potential user question and a question template including the template question "how much is the [Subject_Title]", a plurality of assigned answers related to MSRP, various contract prices, a standalone price, and a shipping cost, respectively, are determined. The question template is associated with the subject "Apple iPhone 6", which is a partial match for "apple iphone". No other matching potential user questions are determined. A response including the determined answers is generated and displayed in a response window **720**.

[0057] FIG. **2** is an example schematic diagram of the content manager **120** according to an embodiment. The content manager **120** includes a processing circuitry **410** coupled to a memory **215**, a storage **220**, and a network interface **230**. In another embodiment, the components of the content manager **120** may be communicatively connected via a bus **240**.

[0058] The processing circuitry **210** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

[0059] The memory **215** may be volatile (e.g., RAM, etc.), non-volatile (e.g., ROM, flash memory, etc.), or a combination thereof. In one configuration, computer readable instructions to implement one or more embodiments disclosed herein may be stored in the storage **220**.

[0060] In another embodiment, the memory **215** is configured to store software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the one or more processors, cause the processing circuitry **210** to perform the various processes described herein. Specifically, the instructions, when executed, cause the processing circuitry **210** to perform an on-demand authorization of access to protected resources, as discussed hereinabove.

[0061] The storage **220** may be magnetic storage, optical storage, and the like, and may be realized, for example, as flash memory or other memory technology, CD-ROM, Digital Versatile Disks (DVDs), or any other medium which can be used to store the desired information. The storage **220** can primarily store question templates to be processed.

[0062] The network interface **230** allows the content manager **120** to communicate with the administrative device **130**, the databases **140**, the user device **150** or a combination of, for the purpose of, for example, receiving template inputs, storing created question templates and assignment data, receiving queries, providing responses, and the like.

[0063] It should be understood that the embodiments described herein are not limited to the specific architecture illustrated in FIG. **2**, and that other architectures may be equally used without departing from the scope of the disclosed embodiments.

[0064] FIG. **3** is a flowchart **300** depicting the method for creating a question template according to an embodiment. In an embodiment, the method may be performed by the content manager **120**, FIG. **1**.

[0065] At S**310**, at least one template question is defined. Each template question is defined based on a set of template question parameters. To this end, in an embodiment, S**310** includes receiving at least one set of template question parameters from, e.g., an administrator device (e.g., the administrator device **130**, FIG. **1**).

[0066] The template question parameters may include, but are not limited to, subject variables, characteristic variables, terms forming a structure of the question (e.g., "who", "what", "when", "where", "why", "how", "is", "are", "the", "does", "can", etc.), terms indicating degree (e.g., "much", "many", "amount", "number", etc.), images or other multimedia content submitted as a query, and any other information utilized to define questions. The subject and characteristic variables represent a subject and a characteristic mentioned in the template question, and may be replaced with particular values to determine a potential user question to be matched to a query.

[0067] As a non-limiting example, a set of template question parameters may be "does %%Subject_Title%% include %%Characteristic_Title%%", where "%%Subject_Titl%%" is the subject variable and "%%Characteristic_Title%%" is the characteristic variable. When comparing the template question defined by such template question parameters to a query mentioning the subject "apple ipad 3" and the characteristic "camera", a potential user question would be "does apple ipad 3 have camera".

[0068] At S**320**, at least one subject assignment is obtained. The at least one subject assignment includes at least one subject to which the created question template will be assigned. In an embodiment, S**320** includes receiving assignment inputs indicating the at least one subject assignment from, e.g., the administrator device.

[0069] At S**330**, a set of at least one answer is assigned to the at least one template question with respect to each assigned subject. Each assigned answer includes content such as, but not limited to, information, pictures, videos, charts, graphs, and the like. In an embodiment, S**330** includes receiving answer inputs indicating the answers to be assigned to the at least one template question from, e.g., the administrator device. The answer inputs may include, but are not limited to, content to be included in each answer, an identifier (e.g., a selection or a location in storage) of content to be included in each answer, and the like. In a further embodiment, S**330** may include obtaining the content for each answer.

[0070] At S**340**, a question template including the defined at least one template question, the answer set assigned to the at least one template question for each subject, and an assignment to the at least one subject is created. In an embodiment, S**340** may include storing the created question template in a storage (e.g., the database **140**, FIG. **1**).

[0071] At optional S**350**, the created question template may be modified. The modifications may include, but are not limited to, adding template questions, deleting template questions, editing template questions, adding answers, combinations thereof, and the like. The modifications may be provided by an administrator, or may be modified based on one or more modification rules.

[0072] Adding the template questions may include, for example, identifying potential template questions to be added based on popularity with respect to the same subject or based on a query for which a matching question template could not be found. Deleting the template questions may include, for example, deleting duplicate questions assigned to the same subject. In an embodiment, S**350** may include sending a notification with proposed modifications and prompting an administrator to accept, reject, or change the proposed modifications.

[0073] At S**360**, it is determined if additional question templates are to be created and, if so, execution continues with S**310**; otherwise, execution terminates. Defining multiple question templates allows for population of a template database for subsequent use.

[0074] It should be noted that the steps of the method illustrated in FIG. **3** are shown being performed sequentially merely for simplicity purposes and without limitation on the disclosed embodiments. In particular,

[0075] FIG. **4** is an example flowchart **400** illustrating a method for generating responses to queries using question templates according to an embodiment. In an embodiment, the method may be performed by the content manager **120**, FIG. **1**. In a further embodiment, the method may be performed based on question templates created as described herein above with respect to FIG. **3**.

[0076] At S**410**, a query is received. The query may include, but is not limited to, text, images, both, and the like. The query may be received from a user device (e.g., the user device **150**, FIG. **1**).

[0077] At S**420**, at least one similar question template is selected based on the query. In an embodiment, S**420** includes identifying terms in the query and comparing the identified terms to terms of each of a plurality of template questions, where each template question is associated with a predetermined question template. The identified terms may include, but are not limited to, structural words (e.g., question words such as "who" and "what", definite articles such as "the", conjunctions such as "and", etc.), words representing a subject, words representing a characteristic, a combination thereof, and the like.

[0078] In an embodiment, S**420** may include comparing the identified terms to predetermined structural words, predetermined subjects, predetermined characteristics, or a combination thereof, of each of the plurality of template questions, and calculating a similarity score for each template question. In a further embodiment, the selected question template may be the question template associated with the template question having the highest similarity score.

[0079] At S**430**, at least one potential user question is generated based on template questions of the at least one similar question template. In a further embodiment, S**430** includes replacing each subject variable in each template question of the at least one similar question template with the identified subject of the query, replacing each characteristic variable in a template question with the identified characteristic of the query, or both, thereby generating a potential user question including a plurality of terms. If a template question does not include any variables (e.g., if the template question is a unique template question lacking variables), the template question may be utilized as one of the at least one potential user question.

[0080] At S**440**, each of the generated at least one potential user question is compared to the query to determine a template question that best matches the query. In an embodiment, S**440** includes comparing each term of the query to each term of each potential user question. In a further embodiment, the comparison is based on one or matching rules. The matching rules for matching between a query and a potential user question may include, but are not limited to, each term of the potential user question being the same as or a synonym of a corresponding term in the query, each term of the potential user question matching a corresponding term in the query except for a misspelling or truncation, the potential user question and the query not being substantively different (e.g., with respect to punctuation, capitalization, short words, and stop words), a combination thereof, and the like. A query and a potential user question may match if the template question parameters of the question are in a different order than that of terms in the query so long as the above-noted matching requirements are met.

[0081] In an embodiment, if multiple potential user questions generated based on different template questions matched the query, the highest ranked template question may be determined. In another embodiment, if no template question matches the query (e.g., if no template question meets the matching rules), S**440** may result in a null value.

[0082] At S**450**, it is determined if the query matches a potential user question and, if so, execution continues with S**460**; otherwise, execution continues with S**470**. In an embodiment, the query does not match a potential user question if S**440** resulted in a null value.

[0083] At S**460**, when it is determined that the query matches a potential user question, a question template for responding to the query is selected. The selected question template is the question template including the best matching template question.

[0084] At S**470**, a response to the query is generated. If an answer set is determined at S**460**, the generated response includes the determined answer set. If no answer set is determined at S**460**, the generated response may include search results obtained by sending the query to a search engine. It should be noted that search results may also be included in the generated response in addition to the determined answer set in order to, e.g., provide supplemental information with the answer set.

[0085] FIG. **8** is an example screenshot **800** illustrating a response to a user query. The example screenshot **800** shows a search bar **810**, a response portion **820**, and a search results portion **830**. The search bar **810** allows a user to input free text queries. In the example screenshot **800**, the user has entered the query "is the apple iphone 6s available". Based on the query, a matching template question is determined, and answers including an image of an iPhone and information regarding availability that are assigned to the matching template question are displayed in the response portion **820**. Additionally, search results from a search engine are displayed in the search results **830**. The example response shown in FIG. **8** may be utilized for, e.g., providing search results as supplemental information for a single direct answer to the user's query.

[0086] FIG. **9** is an example screenshot **900** illustrating a comparison chart provided as a response to a user query. The example screenshot **900** includes a search bar **910**, images **920-1** and **920-2**, and a comparison chart **930**. The search bar **910** allows a user to input free text queries. In the example screenshot **900**, the user has entered the query "compare apple iphone 6s and samsung galaxy 6s". Based on the query, a matching template question is determined, and answers including images **920-1** and **920-2** of an iPhone 6s and a Samsung Galaxy S6, respectively, as well as a comparison chart **930** illustrating a side-by-side comparison of various specifications of the respective phones, are displayed.

[0087] The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPUs"), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

[0088] All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

[0089] It should be understood that any reference to an element herein using a designation such as "first," "second," and so forth does not generally limit the quantity or order of those elements. Rather, these designations are generally used herein as a convenient method of distinguishing between two or more elements or instances of an element. Thus, a reference to first and second elements does not mean that only two elements may be employed there or that the first element must precede the second element in some manner. Also, unless stated otherwise, a set of elements comprises one or more elements.

[0090] As used herein, the phrase "at least one of" followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including "at least one of A, B, and C," the system can include A alone; B alone; C alone; A and B in combination; B and C in combination; A and C in combination; or A, B, and C in combination.

1. A method for handling queries, comprising:

receiving a query;

determining, based on the received query, a best matching template question;

selecting, from among a plurality of question templates, a question template for responding to the query, wherein the selected question template includes the determined best matching template question and a set of answers assigned to the selected question template; and

generating, based on the selected question template, a response to the query.

2. The method of claim **1**, wherein the generated response includes the set of answers assigned to the selected question template.

3. The method of claim **1**, further comprising:

selecting, based on the query, at least one similar question template, each similar question template including at least one template question;

generating, based on the at least one template question of each similar question template, at least one potential user question; and comparing each generated potential user question to the query, wherein the best matching template question is determined based on the comparison.

4. The method of claim **3**, wherein selecting the at least one similar question template further comprises:

identifying a plurality of terms in the received query; and

comparing the identified terms to terms included in each template question of each selected question template, wherein the at least one similar question template is selected based on the comparison.

5. The method of claim **3**, wherein the identified terms include at least one of a subject and a characteristic, wherein each template question includes at least one of a subject variable and a characteristic variable.

6. The method of claim **5**, wherein each generated potential user question further comprises:

replacing each subject variable of each template question of each selected question template with the identified subject term.

7. The method of claim **5**, wherein each generated potential user question further comprises:

replacing each characteristic variable of each template question of each selected question template with the identified characteristic term.

8. The method of claim **3**, wherein the at least one potential user question is the best matching template question when the best matching template question does not include at least one variable.

9. The method of claim **1**, further comprising:

creating the plurality of question templates, wherein creating each question template includes:

receiving a plurality of template inputs, the template inputs including a set of template question parameters;

defining at least one template question based on the set of template question parameters; and

creating the question template based on the received plurality of template inputs, the created question template including the defined at least one template question and a set of answers, wherein the created question template is assigned to at least one subject.

10. A non-transitory computer readable medium having stored thereon instructions for causing one or more processing units to perform a process, the process comprising:

receiving a query;

determining, based on the received query, a best matching template question;

selecting, from among a plurality of question templates, a question template for responding to the query, wherein the selected question template includes the determined best matching template question and a set of answers assigned to the selected question template; and

generating, based on the selected question template, a response to the query.

11. A system for handling queries, comprising:

a processing circuitry; and

a memory, wherein the memory contains instructions that, when executed by the processing circuitry, configure the system to:

receive a query;

determine, based on the received query, a best matching template question;

select, from among a plurality of question templates, a question template for responding to the query, wherein the selected question template includes the determined best matching template question and a set of answers assigned to the selected question template; and

generate, based on the selected question template, a response to the query.

12. The system of claim **11**, wherein the generated response includes the set of answers assigned to the selected question template.

13. The system of claim **11**, wherein the system is further configured to:

select, based on the query, at least one similar question template, each similar question template including at least one template question;

generate, based on the at least one template question of each similar question template, at least one potential user question; and

compare each generated potential user question to the query, wherein the best matching template question is determined based on the comparison.

14. The system of claim **13**, wherein the system is further configured to:

identify a plurality of terms in the received query; and

compare the identified terms to terms included in each template question of each selected question template, wherein the at least one similar question template is selected based on the comparison.

15. The system of claim **13**, wherein the identified terms include at least one of a subject and a characteristic, wherein each template question includes at least one of a subject variable and a characteristic variable.

16. The system of claim **15**, wherein the system is further configured to:

replace each subject variable of each template question of each selected question template with the identified subject term.

17. The system of claim **15**, wherein the system is further configured to:

replace each characteristic variable of each template question of each selected question template with the identified characteristic term.

18. The system of claim **13**, wherein the at least one potential user question is the best matching template question when the best matching template question does not include at least one variable.

19. The system of claim **11**, wherein the system is further configured to, for each of the plurality of question templates:

receive a plurality of template inputs, the template inputs including a set of template question parameters;

define at least one template question based on the set of template question parameters; and

create the question template based on the received plurality of template inputs, the created question template including the defined at least one template question and a set of answers, wherein the created question template is assigned to at least one subject.

* * * * *

EXHIBIT H

US 20180026853A1

(19) United States

(12) Patent Application Publication

SPECTOR

(10) Pub. No.: US 2018/0026853 A1

(43) Pub. Date: Jan. 25, 2018

(54) SYSTEM AND METHOD FOR DETERMINING RESOURCES UTILIZATION IN A VIRTUAL NETWORK

(71) Applicant: **imVision Software Technologies Ltd.**, Ramat Gan (IL)

(72) Inventor: **Yoav SPECTOR**, Tel-Aviv (IL)

(73) Assignee: **imVision Software Technologies Ltd.**, Ramat Gan (IL)

(21) Appl. No.: **15/657,514**

(22) Filed: **Jul. 24, 2017**

Related U.S. Application Data

(60) Provisional application No. 62/366,333, filed on Jul. 25, 2016.

Publication Classification

(51) **Int. Cl.**

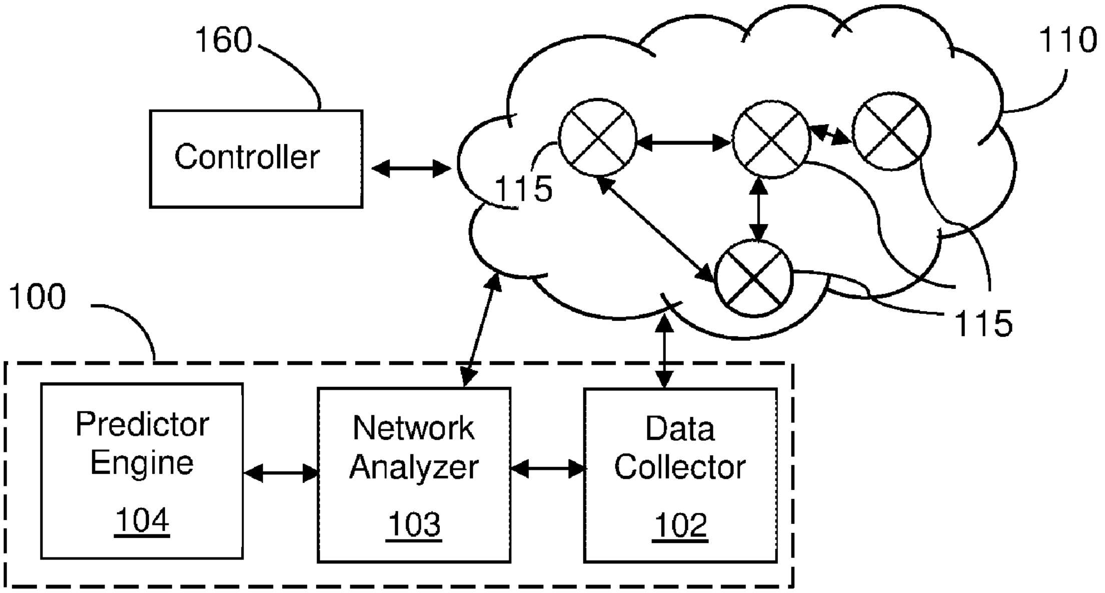
H04L 12/24	(2006.01)
H04L 12/911	(2006.01)
H04L 12/46	(2006.01)
H04L 12/26	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 41/147*** (2013.01); ***H04L 43/0876*** (2013.01); ***H04L 47/70*** (2013.01); ***H04L 12/4641*** (2013.01)

(57) **ABSTRACT**

A system and method for determining optimized resources utilization in a virtual network. The method includes collecting a computing resource parameter from a virtual network, where the computing resource parameter includes at least performance measurements of computing resources in the virtual network; accessing a key performance indicator, where the key performance indicator includes measurement of network traffic performance; and determining, based on the computing resource parameter and the key performance indicator, an optimized resource parameter over a period of time that indicates a usage rule for a computing resource component.



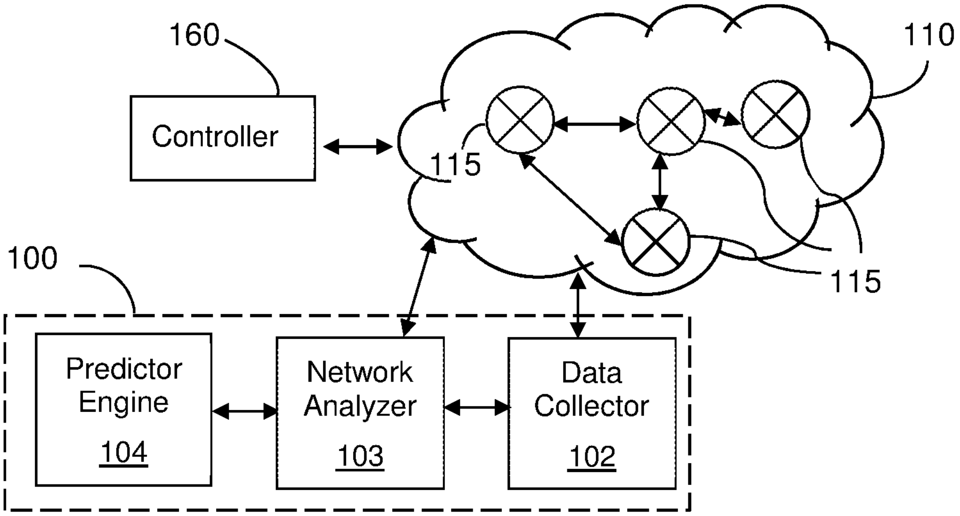

160
Controller
110
115
115
100
Predictor Engine
104
Network Analyzer
103
Data Collector
102

FIG. 1

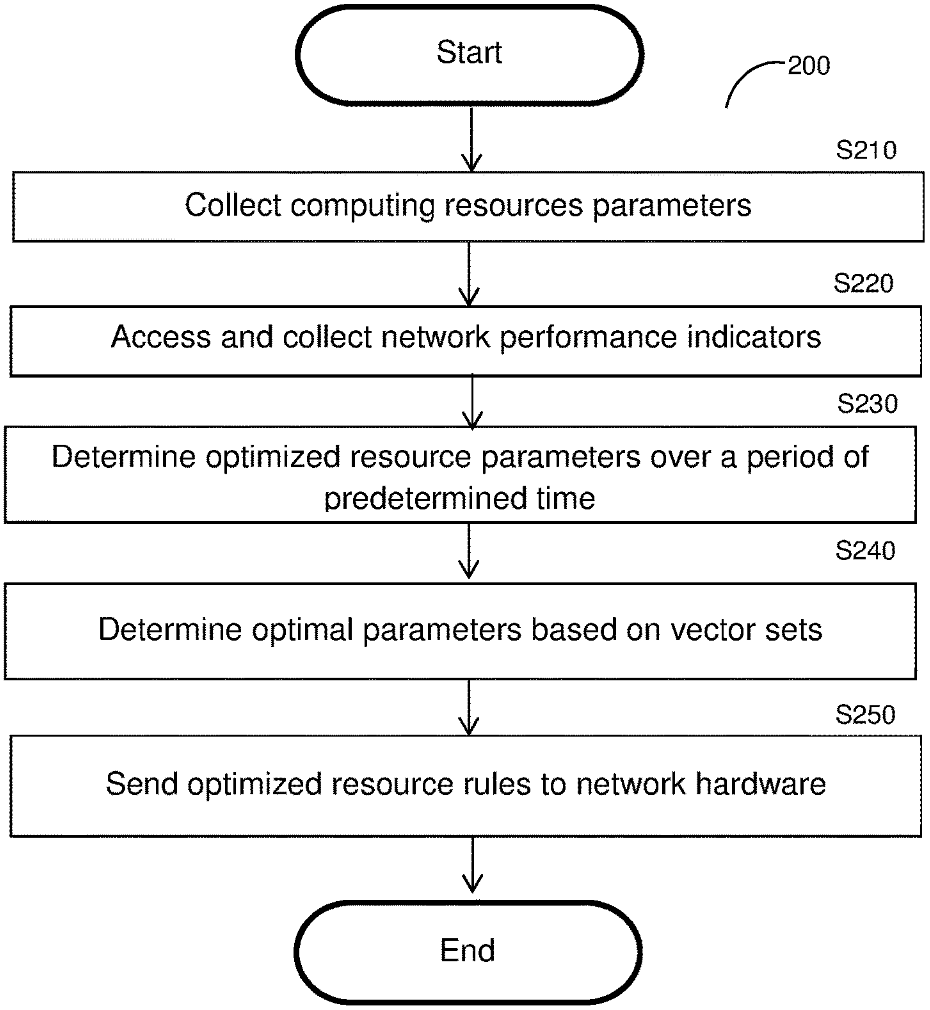

Start
200
S210
Collect computing resources parameters
S220
Access and collect network performance indicators
S230
Determine optimized resource parameters over a period of predetermined time
S240
Determine optimal parameters based on vector sets
S250
Send optimized resource rules to network hardware
End

FIG. 2

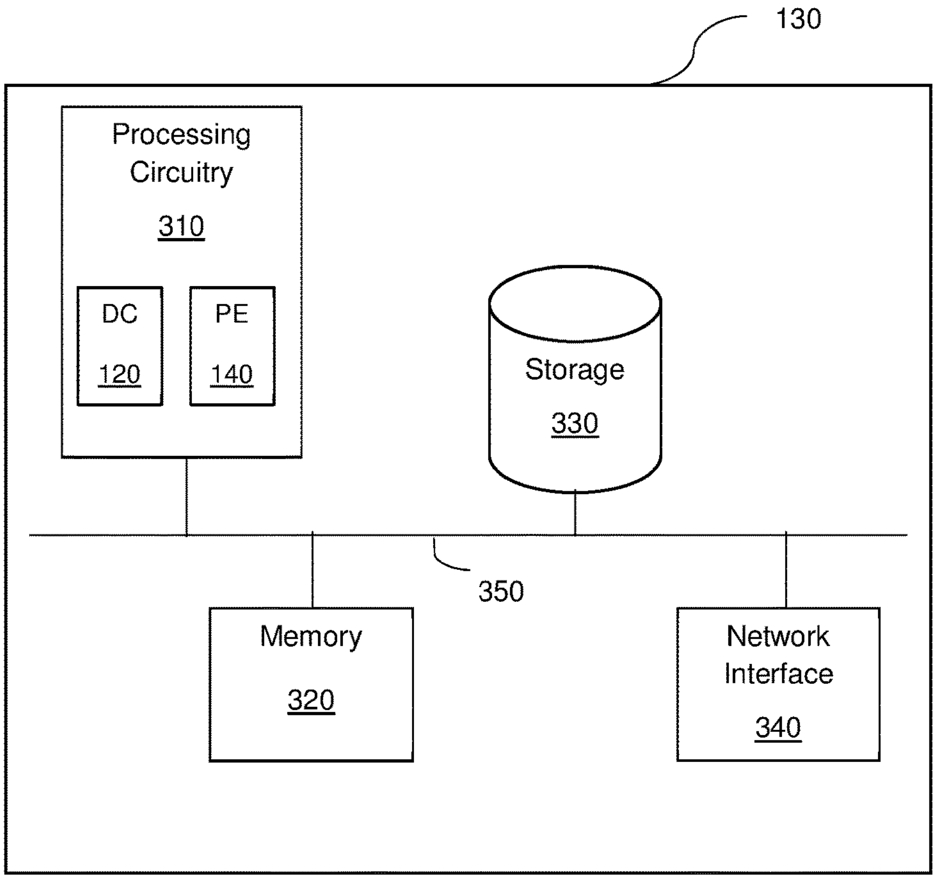

130
Processing Circuitry
310
DC
120
PE
140
Storage
330
350
Memory
320
Network Interface
340

FIG. 3

SYSTEM AND METHOD FOR DETERMINING RESOURCES UTILIZATION IN A VIRTUAL NETWORK

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 62/366,333 filed on Jul. 25, 2016, the contents of which are hereby incorporated by reference.

TECHNICAL FIELD

[0002] The present disclosure relates generally to techniques for determining resource utilization, and more particularly, to techniques for predicting resource utilization for use in virtualized network environments.

BACKGROUND

[0003] The concept of virtualization is currently in widespread commercialized use. A virtual machine is an entity that behaves like a computer, e.g. running an operating system and executing an application, but is separated from the underlying hardware resources used to perform and complete those tasks. A virtual machine is granted access to a portion of the available hardware resources from a hardware system (e.g., CPU, memory, disk space, network communication, etc.). The amount of resources allocated to a particular virtual machine is determined by the computing tasks and applications it is required to perform.

[0004] Another trend that has recently developed is the migration of computing resources (e.g., CPU, memory, disk space, network communication, etc.) from private domains to remotely connected cloud-computing platforms. Applications of various kinds are being executed within such platforms, allowing access to the "economy of scale" environment of cloud-computing platforms.

[0005] A new approach to networking has recently emerged to take advantage of the above-mentioned developments. A software defined network (SDN) allows making communication networks flexible and configurable and enables network engineers and administrators to respond quickly to changing network requirements.

[0006] In an SDN, the network traffic can be controlled from a centralized controller. This can be achieved through commands issued by the controller to the individual switches. Further, in an SDN, services can be delivered to wherever they are needed in a network, without depending on what any particular hardware device is physically connected to within the network. The key technologies involved are functional separation, network virtualization and automation through programmability.

[0007] In a conventional network, a specialized appliance such as a firewall or load balancer is utilized to balance or regulate the traffic. An SDN deploys an application that uses the controller to manage data plane behavior.

[0008] Network functions virtualization (NFV) is a network architecture utilizing methods of information technology (IT) virtualization to virtualize entire classes of network node functions into building blocks that may connect, or chain together, to create communication services.

[0009] Typically, NFV relies upon, but differs from, traditional server-virtualization techniques, such as those used in enterprise IT. A virtualized network function, or VNF are typically based one or more virtual machines running different software and processes, on top of standard high-volume servers, switches, storage devices, or cloud computing platforms, instead of having custom hardware appliances for each network function.

[0010] For example, a virtual session border controller can be deployed to protect a network without the typical cost and complexity of obtaining and installing physical network protection units. Other examples of NFV include virtualized load balancers, virtual firewalls, intrusion detection devices and WAN accelerators.

[0011] Both SDNs and NFV networks often allow major economical savings in network expenditures, as well as offer an efficient and flexible manner of inserting changes into a network. However, since various VNFs running as software tasks can be critical tasks, their performance should be constantly maintained at the same level without any degradation. As such, computing resources utilized by such virtualized environments should be readily available at any moment.

[0012] However, in many virtualized environments other tasks that are non-network related often share the same hardware resources. As such, a load created by such tasks may consume extensive computing resources and lead to the malfunctioning of some or all of the network functions that rely on the same hardware resources.

[0013] It would therefore be advantageous to provide a solution that would overcome the challenges noted above.

SUMMARY

[0014] A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the term "some embodiments" or "certain embodiments" may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

[0015] Certain embodiments disclosed herein include a method for determining optimized resources utilization in a virtual network, the method comprising: collecting at least one computing resource parameter, wherein the at least one computing resource parameter includes at least a measurements of computing resources performance in the virtual network; accessing at least one key performance indicator, where the at least one key performance indicator includes measurement of network traffic performance; determining, based on the at least one computing resource parameter and the at least one key performance indicator, at least one optimized resource parameter over a period of predetermined time interval, wherein the optimized resource parameter indicates a usage rule for a computing resource component.

[0016] Certain embodiments disclosed herein include a non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to perform a method for determining optimized resources utilization in a virtual network, the method comprising: collecting at least one computing resource parameter, wherein the at least one computing resource parameter includes at least measurements of computing resources performance in the virtual network; accessing at least one key performance indicator, where the at least one key performance indicator includes measurement of network traffic performance; determining, based on the at least one computing resource parameter and the at least one key performance indicator, at least one optimized resource parameter over a period of predetermined time interval, wherein the optimized resource parameter indicates a usage rule for a computing resource component.

[0017] Certain embodiments disclosed herein include a system for determining resources utilization in a virtual network, the system comprising: a processing circuitry; and a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: collect at least one computing resource parameter, wherein the at least one computing resource parameter includes at least measurements of computing resources performance in the virtual network; access at least one key performance indicator, where the at least one key performance indicator includes measurement of network traffic performance; determine, based on the at least one computing resource parameter and the at least one key performance indicator, at least one optimized resource parameter over a period of predetermined time interval, wherein the optimized resource parameter indicates a usage rule for a computing resource component

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

[0019] FIG. **1** is a schematic diagram of a network system utilized to describe the various disclosed embodiments.

[0020] FIG. **2** is a flowchart illustrating a method for determining network optimization parameters according to an embodiment.

[0021] FIG. **3** is a block diagram of a network analyzer realized according to an embodiment.

DETAILED DESCRIPTION

[0022] It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

[0023] The various disclosed embodiments include a method and system for determining resources utilization in virtualized network environments. In an embodiment, the method includes collecting data about the status of hardware resources and the status of network performance to determine relevant parameters that are indicative of a restriction or overbearing load of at least one network resource. A set of key performance indicators (KPIs) can be used as a reference to ensure that the network is configured to adequately perform critical tasks within a virtualized network environment.

[0024] According to some embodiments, the disclosed system includes a predictor engine configured to receive parameters regarding the status of computing resources along with the KPIs over a predetermined period of time. The predictor engine is further configured to predict the resources utilization of the virtualized network environment required when executing a critical task.

[0025] FIG. **1** shows an example schematic diagram of a network diagram utilized to describe the various disclosed embodiments. In the deployment illustrated in FIG. **1**, a system **100** is utilized to monitor the performance of a virtual network **110**. In an example embodiment, virtual network **110** can reside on a server or any similar computing device.

[0026] In another example embodiment, the virtual network **110** is a software defined network (SDN). In such deployment, the virtual network **110** includes an optional central controller **160** and a plurality of network nodes **115**. The network nodes **115** communicate with the central controller **160** using, for example, an OpenFlow protocol. The central controller **160** can configure the network nodes **115** to perform certain data path operations. The virtual network **110** can be implemented in wide area networks (WANs), local area networks (LANs), the Internet, metropolitan area networks (MANs), ISP backbones, datacenters, inter-datacenter networks, and the like. Each network node **115** in the SDN may be a router, a switch, a bridge, and so on. Each network node **115** is implemented in software, and its implementation can be run either on the physical hardware equipment provided by the server or the computing device, or on a virtual machine (VM) created virtually within the server. The central controller **160** is configured to set rules and policies which govern the flow of network traffic through the network nodes **115**.

[0027] The system **100** is configured to perform the disclosed embodiments and includes a data collector **102**, a network analyzer **103**, and a predictor engine **104**. In an embodiment, a data collector **102** is connected to the virtual network **110** and configured to access and collect relevant computing resource parameters regarding the status of the computing resources utilized by the network **110**. In an embodiment, the collected computing resource parameters are processed by the network analyzer **103** that is configured to analyze the status of the computing resources that are used to implement the functions of the virtual network **110**.

[0028] The relevant computing resource parameters include, but are not limited to, measurements of current and historical CPU usage, memory usage, and disk space status of the computing environment that is used to implement the virtual network **110**, and used by virtualized network functions to execute network tasks. The network analyzer **103** may be compatible with software stacks, such as, but not limited, OpenStack® Ceilometer, Monasca, and the like.

[0029] In addition to monitoring the usage of hardware computing resources, the network quality can be measured by the network analyzer **103** by accessing and analyzing KPIs of the network **110**. The KPIs may include, but are not limited to, the percentage of network packets lost, the percentage of network procedures that have not been successfully completed, and time taken for the completion of a network task. The KPIs can be represented as real numbers (for example, a number between 0 and 100 for the percentage of packet loss), or as binary numbers representing a pass or a fail for a particular network task.

[0030] The network analyzer **103** is further connected to a predictor engine **104**. In an embodiment, the predictor engine **104** is configured to receive data from the network analyzer **103** and provides estimates of the future performance of the various network nodes **115**. The operation of the system **100** is discussed in more details in FIG. **1**.

[0031] It should be noted that each of the components of the system **100** including the data collector **102**, the data analyzer **103**, the predictor engine **104** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

[0032] In a non-limiting example, the central controller **160** may be software configured to run on a remote server (not shown), where the remote server is additionally used for data analysis. If the data analysis requires heavy hardware resources usage, the network central controller **160** may be unable to adequately perform, resulting in lost or reduced network performance.

[0033] FIG. **2** shows an example flowchart **200** illustrating a method for determining network optimization parameters and predicting the network future performance according to an embodiment.

[0034] At S**210**, the computing resources parameters are collected. In an embodiment, the computing resources parameters are repeatedly collected at predetermined intervals. The interval may be designated as a parameter ΔT, where ΔT can be set to a configurable value. In an embodiment, a lower value of ΔT offers higher accuracy of computing resource parameter measurements, but requires additional computing resources for calculating such measurements. For example, ΔT can be set to measure the computing resource parameters every 15 seconds. Examples for computing resource parameter includes at least one of: current and historical central processing unit usage, memory usage, and disk space usage of the computer resources components.

[0035] At S**220**, the network KPIs are additionally accessed and collected. Examples for network KPIs may include at least one of: percentage of network packets lost over a network connection, percentage of network procedures that have not been successfully completed over a network connection, and time taken for the completion of a network task over a network connection.

[0036] At S**230**, both the computing resource parameters and the network KPIs are obtained and used to generate a data model describing the connection between the network's performance and the measurements of the computing resources. The data model may be generated using supervised or unsupervised machine learning techniques.

[0037] Using this model, the resource parameters that can be most indicative for predicting future performance of various virtual network functions are determined.

[0038] In addition, a set rules for allocation of the computing resources are determined or configured, for example, by the network analyzer **103**. For example, if a remote server is experiencing a bottleneck of memory such that a network controller, e.g. the network controller **160** of FIG. **1**, is unable to perform its tasks because the host of a remote server is simultaneously offering resources to, for example, a data mining task, the parameters provided to a predictor engine may indicate that a rule should be set for the remote server to allocate a predetermined amount of memory, e.g. 4 gigabytes, for exclusive use by the network controller, thus eliminating the memory bottleneck for the network controller.

[0039] At S**240**, a set of vectors are received in order to determine optimal parameters, as described below. In an embodiment, the most indicative resource parameters are determined via a predetermined set of rules. As a non-limiting example, N computing resource parameters (denoted by $P_1, P_2, \ldots, P_N$) are measured over a time interval [0,M*ΔT] at times t=k*ΔT (where k=0, 1, 2, . . . , M) where, for each of these parameters P_i (i=1, 2, . . . , N), its measurements over time are denoted as a vector $\hat{P}_{\hat{1}_{[t]}}$. Similarly, the KPIs of the network are accessed, collected and measured for every ΔT during the time interval [0,M*ΔT].

[0040] In one embodiment, the set of the most indicative parameters (which is a subset of $P_1, P_2, \ldots, P_N$) and denoted as $\overline{P}$, is determined as follows:

[0041] 1. In each measurement of time t (t=k*ΔT where k=0, 1, 2, . . . , M) of the set of KPIs, a binary indicator I(t) is computed. I(t)=1 if and only if the values of the KPIs indicate that the network is performing without degradation. Otherwise, I(t)=0.

[0042] 2. For each parameter P_i (i=1, 2, ... N), an F-Score algorithm in one dimension is applied to the set of tuples (P_i(t), I(t)) t=k*T (where k=0, 1, 2, . . . , M).

[0043] 3. P_i belongs to $\overline{P}$ if and only if its F-score is above a predefined threshold (TH).

[0044] In one embodiment, a decision that the network is performing without degradation at a certain time t, is based on the number of measured KPIs that indicate no degradation in that time t. If this number is above a predefined value, then no degradation is declared.

[0045] In another embodiment, the set of most indicative parameters $\overline{P}$ is determined by applying a PCA (principal component analysis) dimensionality reduction algorithm to the set of vectors P_i(t) (i=1,2, . . . ,N). Other methods for dimensionality reduction such as, but not limited to, a minimum redundancy maximum relevance method, can also be used.

[0046] In yet another embodiment, the set $\overline{P}$ is determined by generating new parameters. These can be, but not limited to, the first and second derivatives of the parameters P_i(t) (i=1,2, . . . ,N) over time.

[0047] The predictor engine **104** is configured to receive inputs of the set of vectors P_i(t) (i=1,2, . . . ,N) for $t \leq t_0$ and the values of the measured KPIs for $t \leq t_0$ (i.e., t is equal to or smaller than t_0) in order to train a classifier.

[0048] In an embodiment, only certain parameters are selected and used to train the classifier. In yet another embodiment those certain parameters can be the set of most indicative parameters selected according to one of the criteria mentioned above, new generated parameters as mentioned above, or a combination thereof. Therefore, the training sequence to the classifier includes the selected set of parameters mentioned above measured at times $t \leq t_0$ (the independent variables) and the set of KPIs measured at times $t \leq t_0$ (the dependent variables).

[0049] In an embodiment, the classifier can be trained with the values of I(t) for $t \leq t_0$ instead of the values of the measured KPIs. The classifier can be, but is not limited to, a decision tree classifier, a decision forest classifier, logistic regression and the like.

[0050] In yet another embodiment, the classifier can be trained with a time shifted version of the dependent variables by m*ΔT units. This enables the system **100** and in particular the predictor engine **104** to predict the future performance of network nodes at m*ΔT time units in the future, based on the availability of computing resources in the present.

[0051] In an embodiment, once the classifier is trained, at any given time $t > t_0$, upon receiving the set of chosen parameters at t, the output of the classifier will be the predicted values of the KPIs (or the indicator I) at time t=t+m*ΔT. In a further embodiment, a logistic regression method may be implemented by the predictor algorithm.

[0052] At optional S**250**, rules for changing the allocation of one or more computing resources are sent to network hardware, e.g. a relevant server in the virtual network **110**. Following the aforementioned example, a rule is sent to the host of the remote server, to set aside the desired amount of a specific hardware resource, such as memory, for exclusive use by a network node (e.g., one of nodes **115** of FIG. **1**).

[0053] FIG. **3** shows an example block diagram of the system **100** implemented according to an embodiment. The system **100** includes a processing circuitry **310** coupled to a memory **320**, a storage **330**, and a network interface **340**. In an embodiment, the components of the system **100** may be communicatively connected via a bus **350**.

[0054] The processing circuitry **310** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

[0055] In another embodiment, the memory **320** is configured to store software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the one or more processors, cause the processing circuitry **310** to perform the various processes described herein. Specifically, the instructions, when executed, cause the processing circuitry **310** to determine resource utilization in virtual networks, determining network optimization parameters and predicting the network future performance. The methods performed by the circuitry **310** can be utilized to perform the operations of the network analyzer **103**, the data collector **102**, the predictor engine **104** as discussed in detail above.

[0056] The storage **330** may be magnetic storage, optical storage, and the like, and may be realized, for example, as flash memory or other memory technology, CD-ROM, Digital Versatile Disks (DVDs), hard-drives, SSD, or any other medium which can be used to store the desired information. The storage **330** may store the network KPIs and communication consumption patterns associated with one or more communications devices.

[0057] The network interface **340** allows the analyzer and the data collector to communicate the central controller **160** and/or various network resources in the virtual network **110** (FIG. **1**). It should be understood that the embodiments described herein are not limited to the specific architecture illustrated in FIG. **3**, and other architectures may be equally used without departing from the scope of the disclosed embodiments.

[0058] That is, the various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPUs"), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

[0059] It should be understood that any reference to an element herein using a designation such as "first," "second," and so forth does not generally limit the quantity or order of those elements. Rather, these designations are generally used herein as a convenient method of distinguishing between two or more elements or instances of an element. Thus, a reference to first and second elements does not mean that only two elements may be employed there or that the first element must precede the second element in some manner. Also, unless stated otherwise, a set of elements comprises one or more elements.

[0060] As used herein, the phrase "at least one of" followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including "at least one of A, B, and C," the system can include A alone; B alone; C alone; A and B in combination; B and C in combination; A and C in combination; or A, B, and C in combination.

[0061] All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

What is claimed is:

1. A method for determining optimized resources utilization in a virtual network, comprising:

collecting at least one computing resource parameter, wherein the at least one computing resource parameter comprises a measurement of computing resources performance in the virtual network;

accessing at least one key performance indicator, where the at least one key performance indicator includes measurement of network traffic performance;

determining, based on the at least one computing resource parameter and the at least one key performance indicator, at least one optimized resource parameter over a period of predetermined time interval, wherein the optimized resource parameter indicates a usage rule for a computing resource component.

2. The method of claim **1**, further comprising:

configuring the computing resource component with the determined usage rule.

3. The method of claim **1**, wherein the at least one computing resource parameter includes at least one of: current and historical central processing unit usage, memory usage, and disk space usage of the computer resources components.

4. The method of claim **1**, wherein the at least one key performance indicator includes at least one of: percentage of network packets lost over a network connection, percentage of network procedures that have not been successfully completed over a network connection, and time taken for the completion of a network task over a network connection.

5. The method of claim **1**, wherein the at least one computing resource parameter and the at least one key performance indicator are repeatedly collected at predetermined time intervals.

6. The method of claim **1**, further comprising:

generating a data model, based on the at least one computing resource parameter and the at least one key performance indicator, wherein the data model describes a connection between the virtual network's performance and the collected measurements of the at least one computing resource parameter and the at least one key performance indicator.

7. The method of claim **6**, wherein the data model is further configured to determine which of the computing resource parameters and key performance indicators are most indicative of future network performance.

8. The method of claim **1**, further comprising:

predicting future values of the at least one key performance indicator for determining the usage rule.

9. The method of claim **1**, wherein the at least one optimized resource parameter is determined by a dimensionality reduction process.

10. A non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to perform a method for determining optimized resources utilization in a virtual network, the method comprising:

collecting at least one computing resource parameter, wherein the at least one computing resource parameter comprises a measurement of computing resources performance in the virtual network;

accessing at least one key performance indicator, where the at least one key performance indicator includes measurement of network traffic performance;

determining, based on the at least one computing resource parameter and the at least one key performance indicator, at least one optimized resource parameter over a period of predetermined time interval, wherein the optimized resource parameter indicates a usage rule for a computing resource component.

11. A system for determining optimized resources utilization in a virtual network, comprising:

a processing circuitry; and

a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to:

collect at least one computing resource parameter, wherein the at least one computing resource parameter comprises a measurement of computing resources performance in the virtual network;

access at least one key performance indicator, where the at least one key performance indicator includes measurement of network traffic performance;

determine, based on the at least one computing resource parameter and the at least one key performance indicator, at least one optimized resource parameter over a period of predetermined time interval, wherein the optimized resource parameter indicates a usage rule for a computing resource component.

12. The system of claim **11**, further comprising:

configure the computing resource component with the determined usage rule.

13. The system of claim **11**, wherein the at least one computing resource parameter includes at least one of: current and historical central processing unit usage, memory usage, and disk space usage of the computer resources components.

14. The system of claim **11**, wherein the at least one key performance indicator includes at least one of: percentage of network packets lost over a network connection, percentage of network procedures that have not been successfully completed over a network connection, and time taken for the completion of a network task over a network connection.

15. The system of claim **11**, wherein the at least one computing resource parameter and the at least one key performance indicator are repeatedly collected at predetermined time intervals.

16. The system of claim **11**, further comprising:

generating a data model, based on the at least one computing resource parameter and the at least one key performance indicator, wherein the data model describes a connection between the virtual network's performance and the collected measurements of the at least one computing resource parameter and the at least one key performance indicator.

17. The system of claim **16**, wherein the data model is further configured to determine which of the computing resource parameters and key performance indicators are most indicative of future network performance.

18. The system of claim **11**, further comprising:

predict future values of the at least one key performance indicator for determining the usage rule.

19. The system of claim **11**, wherein the at least one optimized resource parameter is determined by a dimensionality reduction process.

* * * * *

EXHIBIT I

STORAGE

This article is more than **1 year old**

Aptare: Eight exabyte-juggler pimps its 'data centre MRI' product

Eight? Really?

Chris Mellor

Thu 6 Jul 2017 // 08:37 UTC

ANALYSIS Data centre infrastructure management console seller Aptare claims to have 8 or 9 exabytes of storage under management – a larger total than any storage company on the planet.

It also claims to make the only DC infra management console you'll likely need, which is quite a claim.

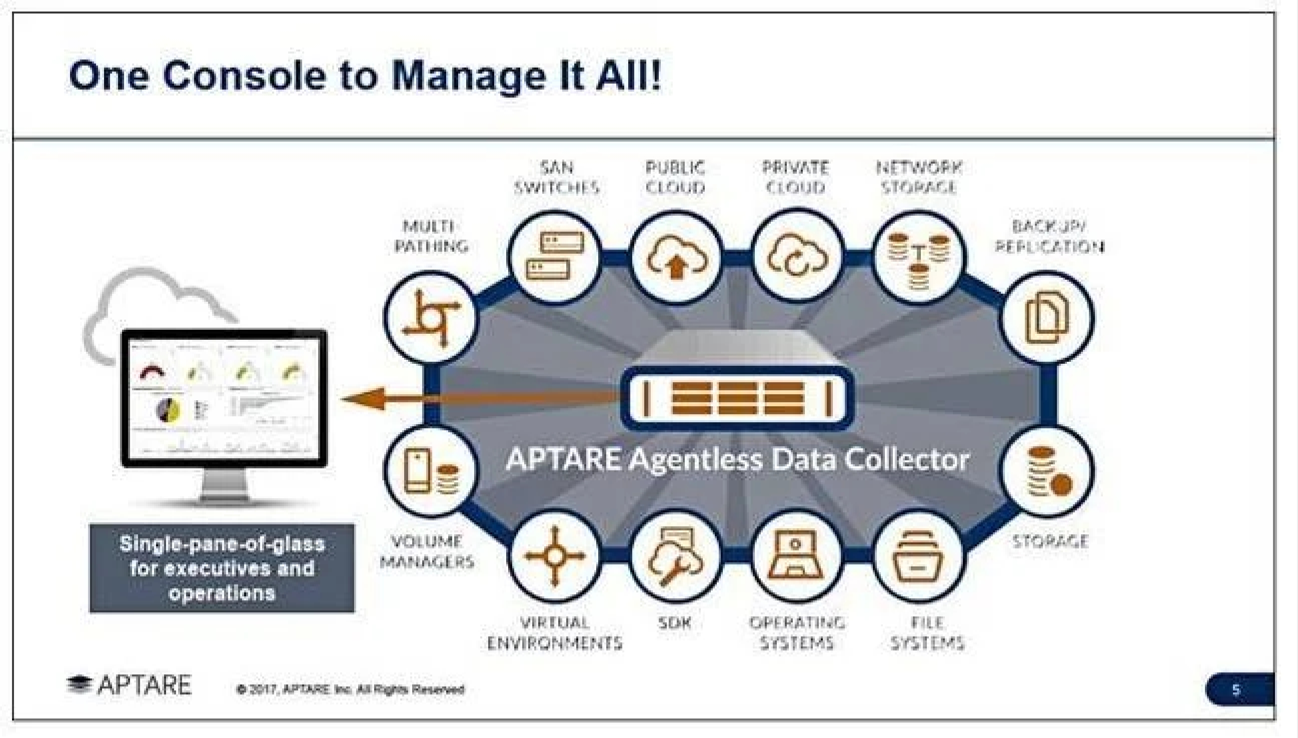
We first came across Aptare in the storage resource management reporting era of 2009-2010 along with Bocada and others. It added SAN fabric component coverage in June 2010, and then went off our radar. Now it's back.

The firm briefed an IT Press Tour in Silicon Valley in June and said it was still privately owned, by its near 100 employees actually, and is profitable with almost 1,000 customers*, most of them very large. This needs stressing: it claims a full 35 per cent of Global Fortune 100 companies are its customers, and some of them run millions of backup jobs a week across their global IT estates.

For example, CitiGroup is a customer and has 60 data centres world-wide, each with an Aptare collector pumping data to a central site.

We learnt that roughly one-third of its business comes through HDS, with whom it has an OEM deal; HDS' Hitachi Storage Reporter product uses Aptare's StorageConsole 6.5 platform.

Aptare's core perception is that you can't manage a data centre infrastructure unless you have an end-to-end view of how its components interact and can present that. It does this with the Aptare console, an agentless data collector from storage repositories, file systems, volume managers, operating systems, SAN switches, backup products, public clouds, and virtual environments.



Each data centre can run the Aptare software, feeding results to a central portal.

The company generates and provides infrastructure information to operations, admin and executive staff, but doesn't automatically take any action based on the information it gathers. Its software can invoke scripts in third-party tools to move data or take other actions, but its own software doesn't do anything itself except gather component-level information, analyse it and present the results.

Capabilities

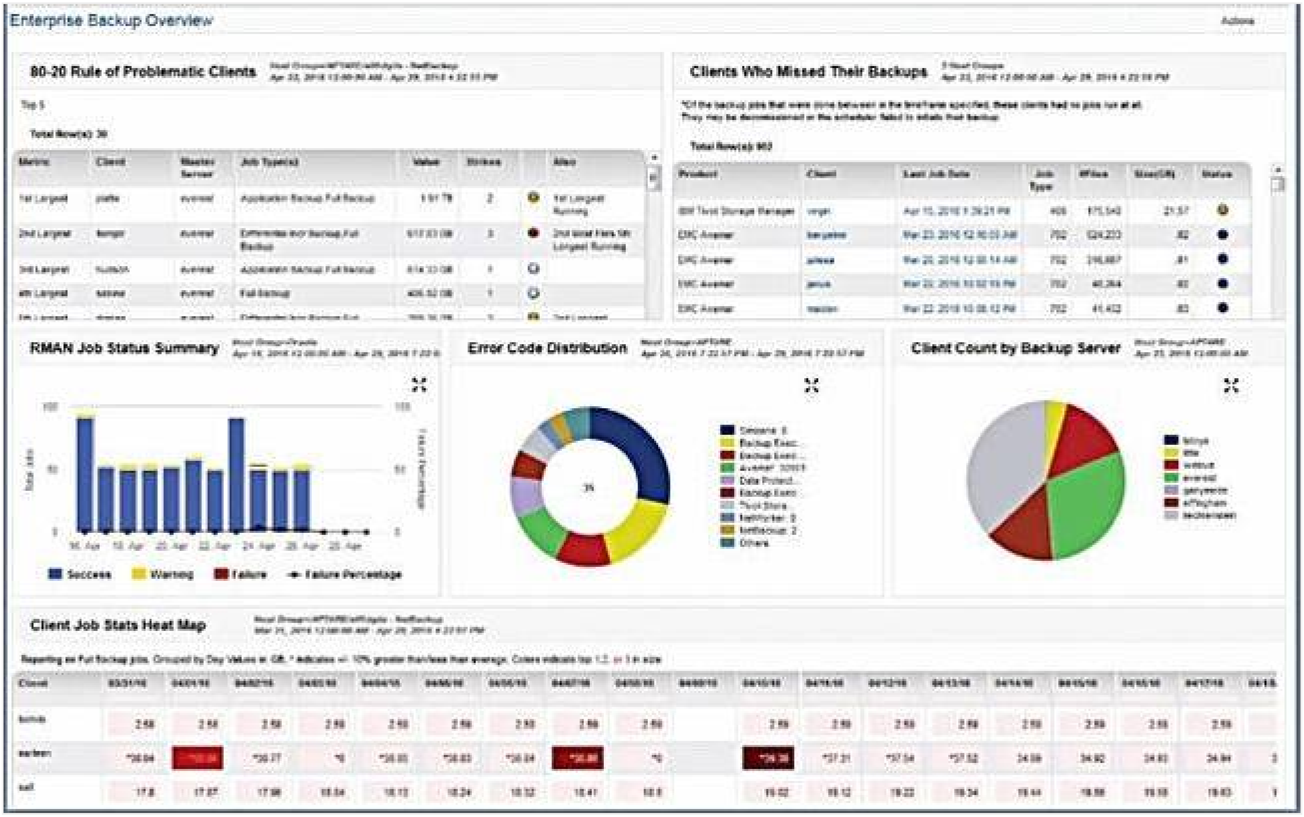
Nordea Bank is an Aptare customer, and has offices in Denmark, London, New York, Poland, Frankfurt and Singapore. It uses Oracle, Veritas and Tivoli applications. Claus Mertz manages its backup operations and says that, with millions of backup jobs, it's impossible to trouble-shoot every failure. He needs and gets root cause and problem isolation analytics, accessed through a single pane of glass independent of the data protection products.

Mertz gets automated standard reporting and spends less time looking for ad-hoc information.

Basically Aptare picks up sensor information from the IT infrastructure components and builds a database, using an Oracle RDBMS, of metadata – information which can be cut, sliced and diced to provide high-level summaries or detailed drill-down exposures through dashboards. For example, Microsoft Azure virtual machine summary, Azure storage account listing, backup server media summary, and Tivoli backup master server overview.

The reasons customers use Aptare's SW includes finding unprotected data, storage optimisation, finding out which people and business units are using which infrastructure, regulatory compliance, SLA adherence, and ad hoc queries answering questions such as: "Is there orphaned capacity which can be reclaimed?"

Aptare says one large multi-national bank famously reclaimed 8PB of unused storage which was costing it $22m two weeks after starting with Aptare's console SW.


Enterprise Backup Overview
80-20 Rule of Problematic Clients
Clients Who Missed Their Backups
RMAN Job Status Summary
Error Code Distribution
Client Count by Backup Server
Success
Warning
Failure
Failure Percentage
Others
Client Job Stats Heat Map

Aptare Console backup overview

Custom reports can built, using drag-and-drop to identify elements for the reports. There is an SDK for new data sources to be added.

Clark said: "Customers can opt in to share their performance profiles to Aptare.com for segments of their storage infrastructure. We retransmit anonymous info about performance profiles for performance comparisons and help spread best practices."

Founding

Aptare founder, CEO and President Rick Clark

Aptare is no startup; it's 24 years old, having been founded in 1993, by Rick Clark, amongst others. Clark said he took out loans and raided his children's college funds to find and risk the money starting up Aptare. There was no VC funding – the company being self-funded from the beginning.

This also meant that it could go its own way, without backseat-driving VCs trimming and changing the course *en route* with their own exit in mind.

Aptare has grown slowly and quietly, compared to a successful VC-funded firm which needs to make marketing noise to help it get attention from customers. Clark's firm has grown more by word of mouth than most.

What's its main strength? Clark says it's the depth and breadth of data gathered, stitched together and analysed: "No one provides the end-to-end view that we do. ... We're the MRI scan - the newer generic tools are just X-rays."

An example: A customer's misconfigured backup job ran at 3pm instead of 3am and killed storage performance. The storage product stats couldn't show any reason for lousy array performance in the afternoon. But Aptare's software, which looks at backup schedules, could and the problem was simply fixed.

Another: a bank found a test and dev backup retention policy had been mistakenly set to "Infinite" and so millions of dollars were spent sending useless tapes offline to Iron Mountain for safe-keeping. Fixing the problem, once found, was ridiculously simple.

Aptare's software can run in Oracle's cloud and the business has a strategic relationship with Oracle, with an embedded software licence agreement. It's not yet listed on the Oracle Cloud.

Clark said: "We don't provide our own public cloud-based service. ... We're not in the Amazon Markeplace yet." Is this a hint that Aptare Console in AWS is coming for customers to use?

Asked how Aptare works with other vendors, Clark said: "EMC and NetApp have their own stuff. Infinidat doesn't and we partner... [But] Dell EMC uses our product to conduct storage assessments specifically around storage reclamation for their customers."

Why is it briefing the European press in the States now? The answer is simple enough; it wants to increase its European sales. Pay attention, value-added resellers. ®

***** Back in 2010 it had 450 customers.

EXHIBIT J

US010127335B2

(12) **United States Patent**
Warshavsky et al.

(10) **Patent No.: US 10,127,335 B2**
(45) **Date of Patent: *Nov. 13, 2018**

(54) **SYSTEM AND METHOD OF PERFORMING ANALYTICS WITH RESPECT TO CONTENT STORING SERVERS CACHING POPULAR CONTENT**

(71) Applicant: **Qwilt, Inc.**, Redwood City, CA (US)

(72) Inventors: **Arnon Warshavsky**, Ben Ammi (IL); **Oren Shemesh**, Moledet (IL); **Gaash Hazan**, Kfra Saba (IL); **Yoav Gressel**, Netaim (IL); **Dan Sahar**, San Francisco, CA (US); **Alon Maor**, Palo Alto, CA (US)

(73) Assignee: **Qwilt, Inc**, Redwood City, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 525 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **14/665,884**

(22) Filed: **Mar. 23, 2015**

(65) **Prior Publication Data**

US 2015/0193567 A1 Jul. 9, 2015

Related U.S. Application Data

(63) Continuation-in-part of application No. 13/006,785, filed on Jan. 14, 2011, now Pat. No. 9,723,073.

(Continued)

(51) **Int. Cl.**

G06F 17/50	(2006.01)
H04L 12/26	(2006.01)

(Continued)

(52) **U.S. Cl.**
CPC ***G06F 17/5009*** (2013.01); ***H04L 29/08729*** (2013.01); ***H04L 43/062*** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC H04L 29/08729; H04L 29/08774; H04L 29/0881; H04L 29/08891; H04L 67/1095;
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,829,046 A	10/1998	Tzelnic et al.
5,893,140 A	4/1999	Vahalia et al.

(Continued)

OTHER PUBLICATIONS

Non-Final Office Action dated Sep. 25, 2012 for U.S. Appl. No. 13/006,785.

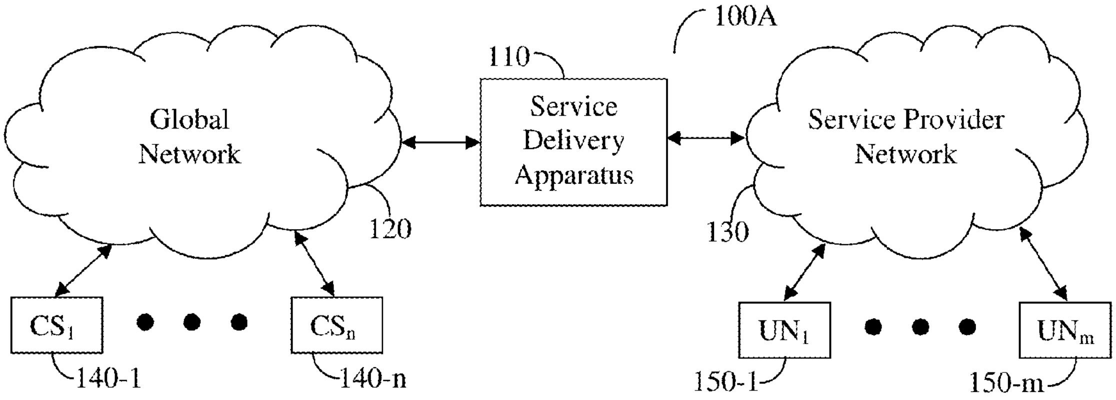
(Continued)

Primary Examiner — Davoud A Zand
(74) *Attorney, Agent, or Firm* — M&B IP Analysts, LLC

(57) **ABSTRACT**

Methods and systems are provided for generating a traffic simulation respective of at least one content storing server, the content storing server operative for caching popular content. One method includes sniffing traffic between at least a first portion of a network and at least a second portion of the network, identifying, from the sniffed traffic, at least a source of content and at least a destination of the content, determining if the content is stored on a cache, simulating a notification respective of the content, and generating a simulation of traffic respective at least of: the content and the simulated notification.

8 Claims, 7 Drawing Sheets



Related U.S. Application Data

(60) Provisional application No. 61/969,418, filed on Mar. 24, 2014, provisional application No. 61/375,836, filed on Aug. 22, 2010.

(51) **Int. Cl.**
H04L 29/08 (2006.01)
H04L 29/06 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 67/06*** (2013.01); ***H04L 67/1095*** (2013.01); ***H04L 67/2842*** (2013.01); ***H04L 67/38*** (2013.01)

(58) **Field of Classification Search**
CPC H04L 67/1097; H04L 67/2847; H04L 67/2842; H04L 67/2852; H04L 67/06; H04L 67/38; H04L 43/062; G06F 17/5009
USPC 709/223, 224; 340/1.1–16.1; 370/252; 714/39, 47.1–47.3
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,933,603	A	8/1999	Vahalia et al.	
5,944,789	A	8/1999	Tzelnic et al.	
5,948,062	A	9/1999	Tzelnic et al.	
6,049,530	A	4/2000	Petersen et al.	
6,061,504	A	5/2000	Tzelnic et al.	
6,363,413	B2	3/2002	Kidder	
6,473,794	B1 *	10/2002	Guheen	H04L 41/22 709/223
6,536,037	B1 *	3/2003	Guheen	G06F 8/71 703/2
6,615,166	B1 *	9/2003	Guheen	G06Q 10/06 703/26
6,700,889	B1	3/2004	Nun	
6,772,193	B1	8/2004	Igawa et al.	
6,799,248	B2	9/2004	Scherr	
6,823,401	B2 *	11/2004	Feather, Jr.	H04L 29/06 710/15
6,831,893	B1	12/2004	Ben-Nun et al.	
6,873,600	B1 *	3/2005	Duffield	H04L 12/2697 370/252
6,985,956	B2 *	1/2006	Luke	G06F 11/2089 709/216
6,986,018	B2	1/2006	O'Rourke et al.	
7,149,698	B2 *	12/2006	Guheen	G06Q 50/01 705/319
7,281,260	B2	10/2007	Puente et al.	
7,310,480	B2	12/2007	Maciocco et al.	
7,349,979	B1 *	3/2008	Cieslak	H04L 67/2814 709/238
7,436,830	B2	10/2008	Ben-Nun et al.	
7,596,664	B2	9/2009	Ishikawa et al.	
7,606,314	B2	10/2009	Coleman et al.	
7,685,254	B2	3/2010	Pandya	
7,697,557	B2	4/2010	Segel	
7,719,966	B2	5/2010	Luft et al.	
7,818,402	B1	10/2010	Zhang	
7,912,921	B2	3/2011	O'Rourke et al.	
7,957,396	B1	6/2011	Kohn et al.	
8,009,682	B2 *	8/2011	Gopinath	G06F 9/5005 370/252
8,065,559	B2	11/2011	Kamath et al.	
8,607,166	B2	12/2013	Jalon et al.	
8,621,101	B1 *	12/2013	Starr	H04L 67/28 709/223
8,706,900	B2	4/2014	Carver et al.	
8,737,407	B2 *	5/2014	Shetty	G06F 9/505 370/395.2
8,937,942	B1 *	1/2015	Li	H04L 67/1097 370/386
2001/0049732	A1	12/2001	Raciborski et al.	
2001/0051980	A1	12/2001	Raciborski et al.	
2002/0006124	A1	1/2002	Jimenez et al.	
2002/0040366	A1	4/2002	Lahr	
2002/0040404	A1	4/2002	Lahr	
2002/0042817	A1	4/2002	Lahr	
2002/0048269	A1	4/2002	Hong et al.	
2002/0062372	A1	5/2002	Hong et al.	
2002/0136204	A1	9/2002	Chen et al.	
2003/0097443	A1	5/2003	Gillett et al.	
2003/0221127	A1	11/2003	Risan et al.	
2004/0128693	A1	7/2004	Weigand	
2004/0133776	A1	7/2004	Putzolu	
2004/0181579	A1	9/2004	Huck et al.	
2005/0015702	A1 *	1/2005	Shier	G06F 11/2273 714/776
2005/0022237	A1	1/2005	Nomura	
2005/0213514	A1 *	9/2005	Su	H04L 41/142 370/254
2005/0289111	A1	12/2005	Tribble et al.	
2006/0129697	A1	6/2006	Vange et al.	
2006/0168318	A1	7/2006	Twiss	
2006/0224687	A1	10/2006	Popkin et al.	
2006/0271972	A1	11/2006	Pai et al.	
2007/0050686	A1 *	3/2007	Keeton	G06F 11/261 714/48
2007/0124781	A1	5/2007	Casey et al.	
2007/0192474	A1	8/2007	Decasper et al.	
2007/0244987	A1	10/2007	Pedersen et al.	
2008/0010381	A1	1/2008	Barraclough et al.	
2008/0307343	A1	12/2008	Robert et al.	
2009/0083279	A1	3/2009	Hasek	
2009/0119734	A1	5/2009	Deshpande et al.	
2009/0172565	A1	7/2009	Jackson et al.	
2009/0193129	A1	7/2009	Agarwal et al.	
2009/0307757	A1	12/2009	Groten	
2009/0313437	A1	12/2009	Sofman et al.	
2010/0023726	A1 *	1/2010	Aviles	G06F 12/0813 711/216
2010/0054257	A1	3/2010	Dolganow et al.	
2010/0082774	A1	4/2010	Pitts	
2010/0115072	A1	5/2010	Payyappilly et al.	
2010/0287227	A1 *	11/2010	Goel	H04L 67/1002 709/202
2011/0055386	A1	3/2011	Middleton et al.	
2011/0078343	A1	3/2011	Resch et al.	
2011/0107185	A1	5/2011	Grube et al.	
2011/0141887	A1	6/2011	Klein et al.	
2011/0153937	A1	6/2011	Annamalaisami et al.	
2012/0011271	A1 *	1/2012	Zhao	H04W 4/18 709/234
2012/0030212	A1	2/2012	Koopmans et al.	
2012/0047224	A1	2/2012	Shemesh et al.	
2012/0047252	A1	2/2012	Maor et al.	
2012/0057460	A1	3/2012	Hussain et al.	
2012/0084464	A1	4/2012	Cochinwala et al.	
2012/0117610	A1	5/2012	Pandya	
2012/0124618	A1	5/2012	Ruiz-Velasco et al.	
2012/0159329	A1	6/2012	Chow et al.	
2012/0159558	A1	6/2012	Whyte et al.	
2014/0108586	A1	4/2014	Zhao	
2016/0021057	A1	1/2016	Frost et al.	

OTHER PUBLICATIONS

Non-Final Office Action dated Jul. 26, 2012 in U.S. Appl. No. 13/006,875.

* cited by examiner

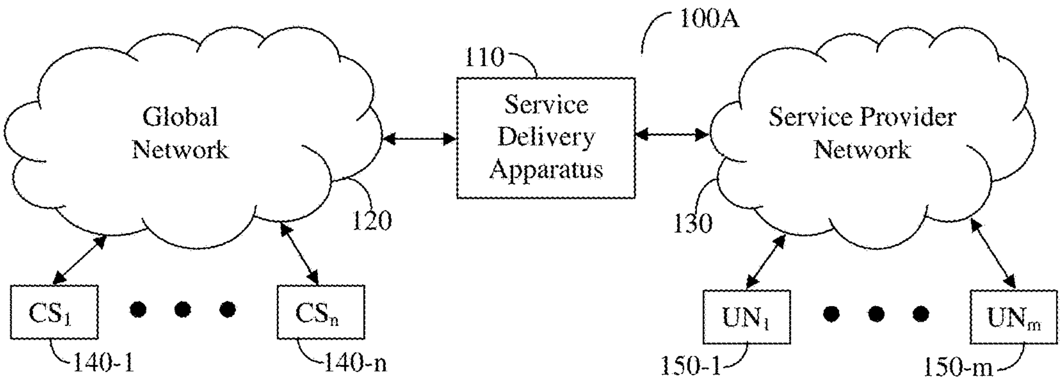


FIGURE 1A

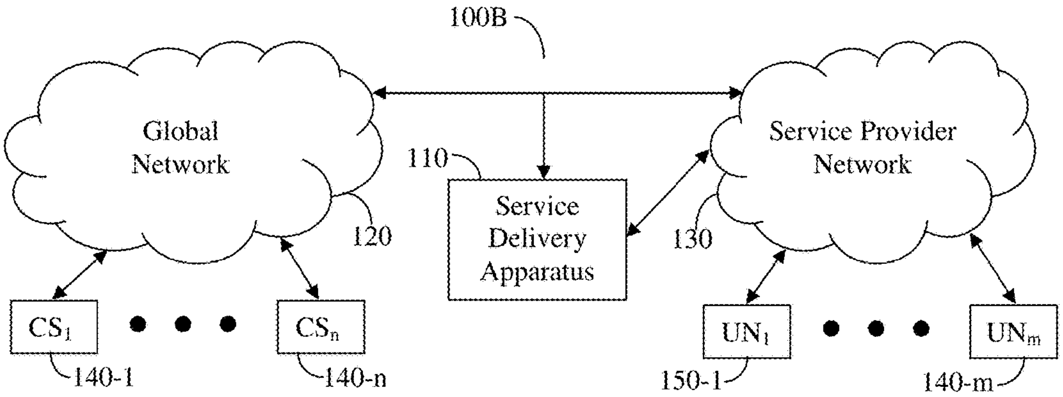


FIGURE 1B

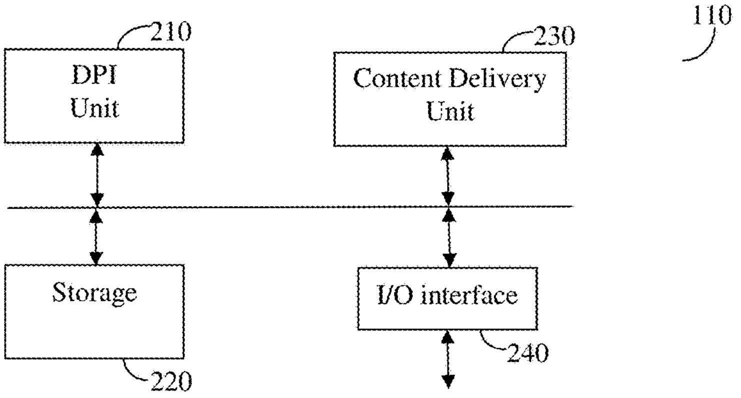


FIGURE 2

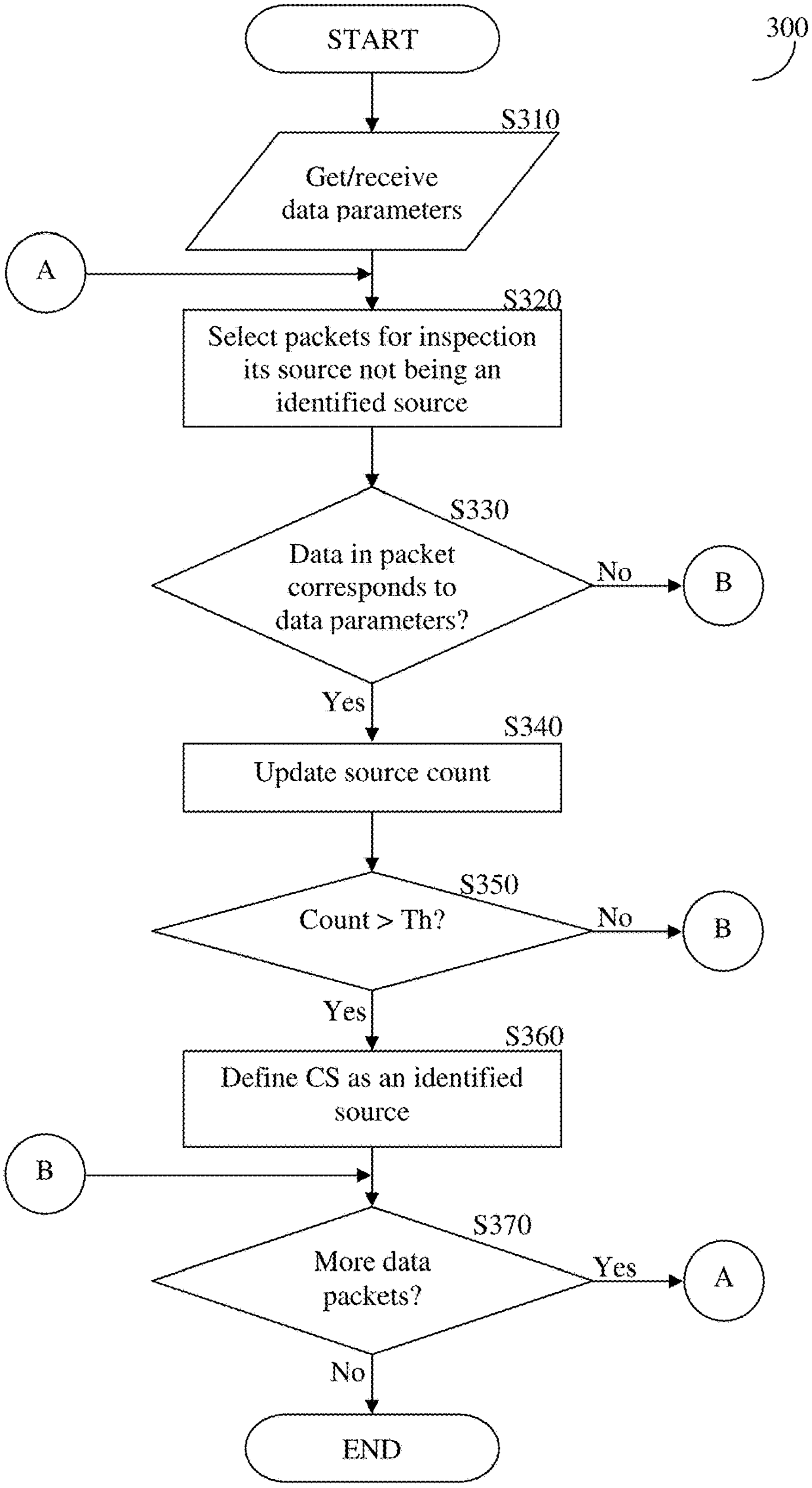

300
START
S310
Get/receive data parameters
A
S320
Select packets for inspection its source not being an identified source
S330
Data in packet corresponds to data parameters?
No
B
Yes
S340
Update source count
S350
Count > Th?
No
B
Yes
S360
Define CS as an identified source
B
S370
More data packets?
Yes
A
No
END

FIGURE 3

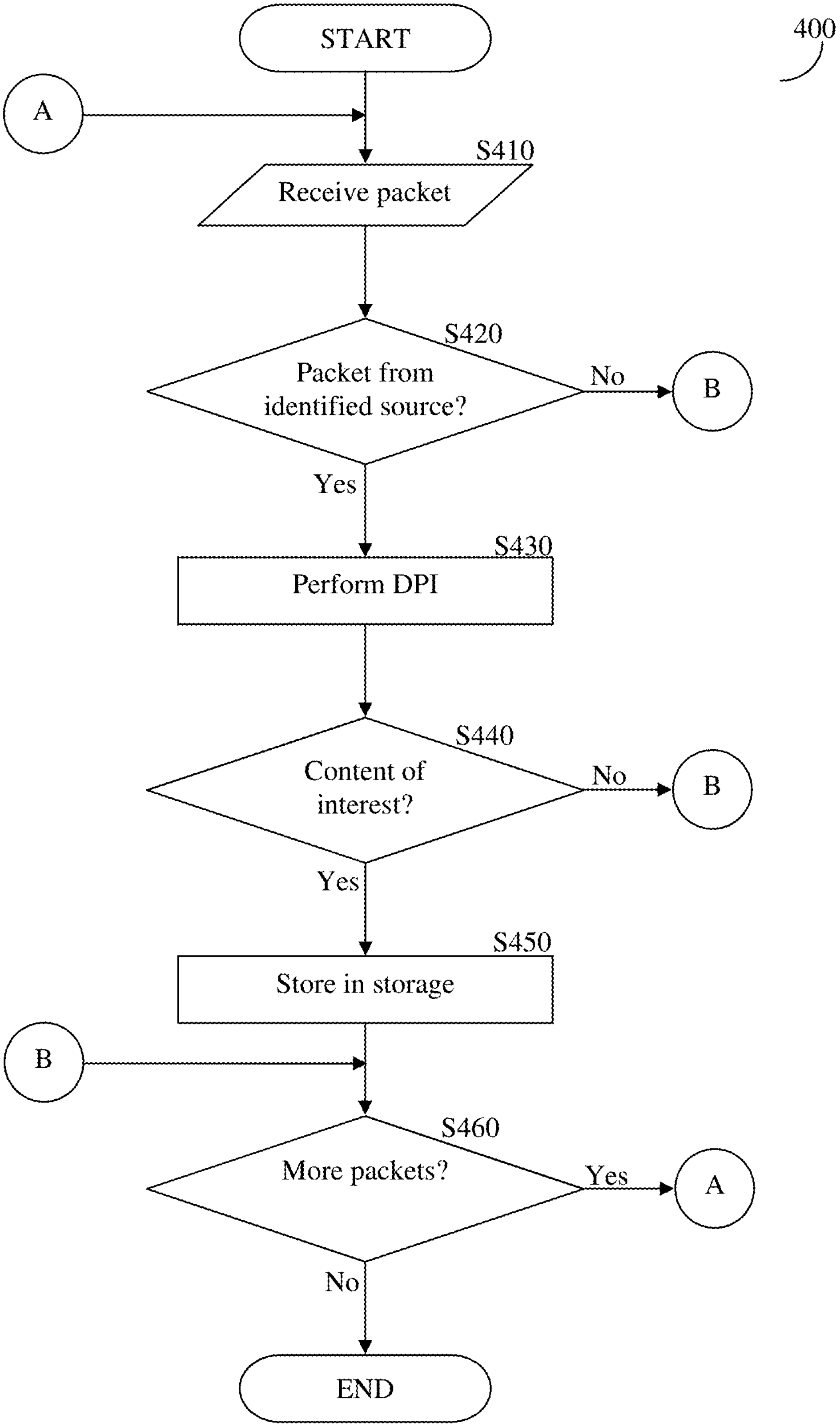

400
START
A
S410
Receive packet
S420
Packet from identified source?
No
B
Yes
S430
Perform DPI
S440
Content of interest?
No
B
Yes
S450
Store in storage
B
S460
More packets?
Yes
A
No
END

FIGURE 4

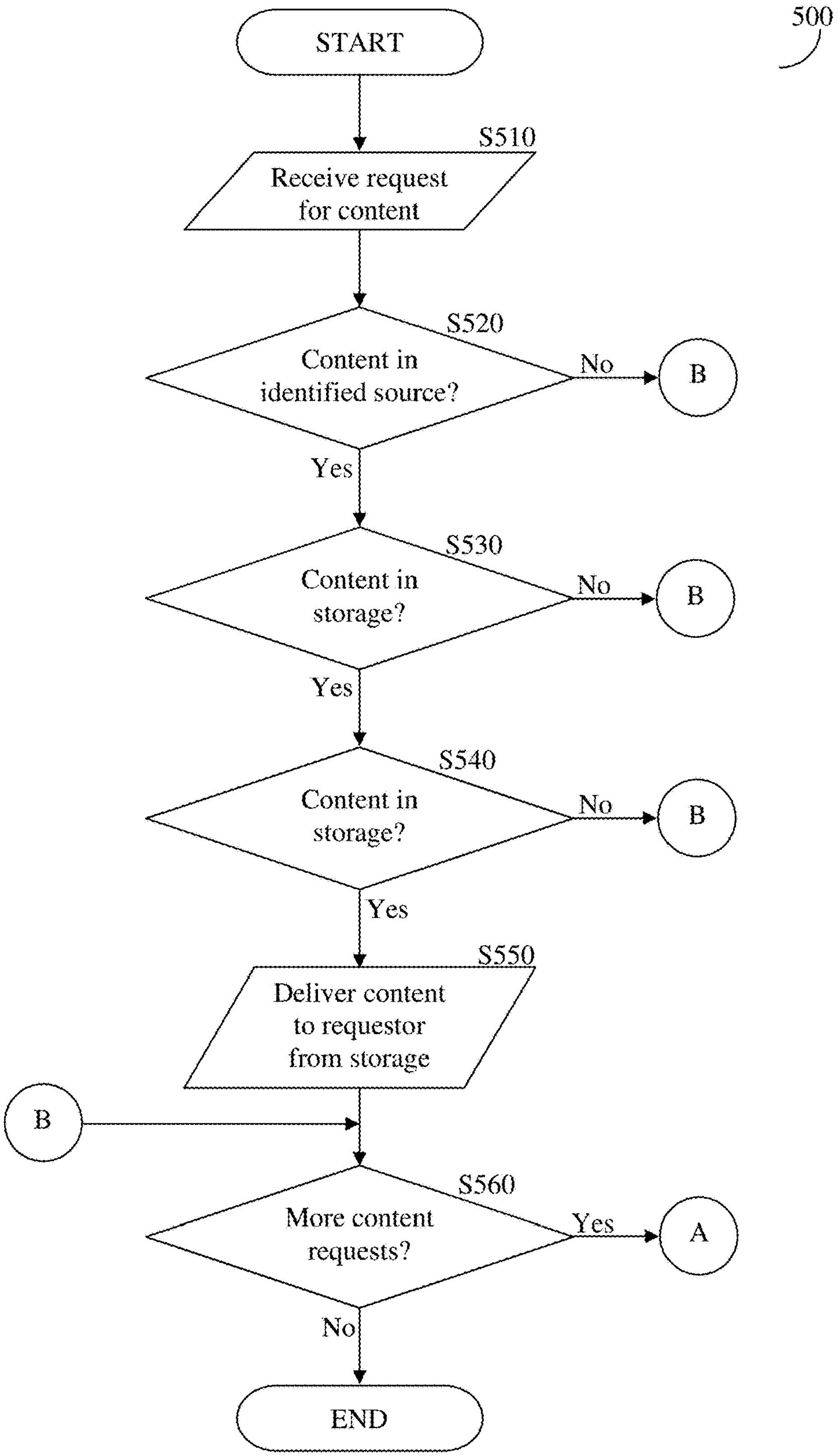

500
START
S510
Receive request for content
S520
Content in identified source?
No
B
Yes
S530
Content in storage?
No
B
Yes
S540
Content in storage?
No
B
Yes
S550
Deliver content to requestor from storage
B
S560
More content requests?
Yes
A
No
END

FIGURE 5

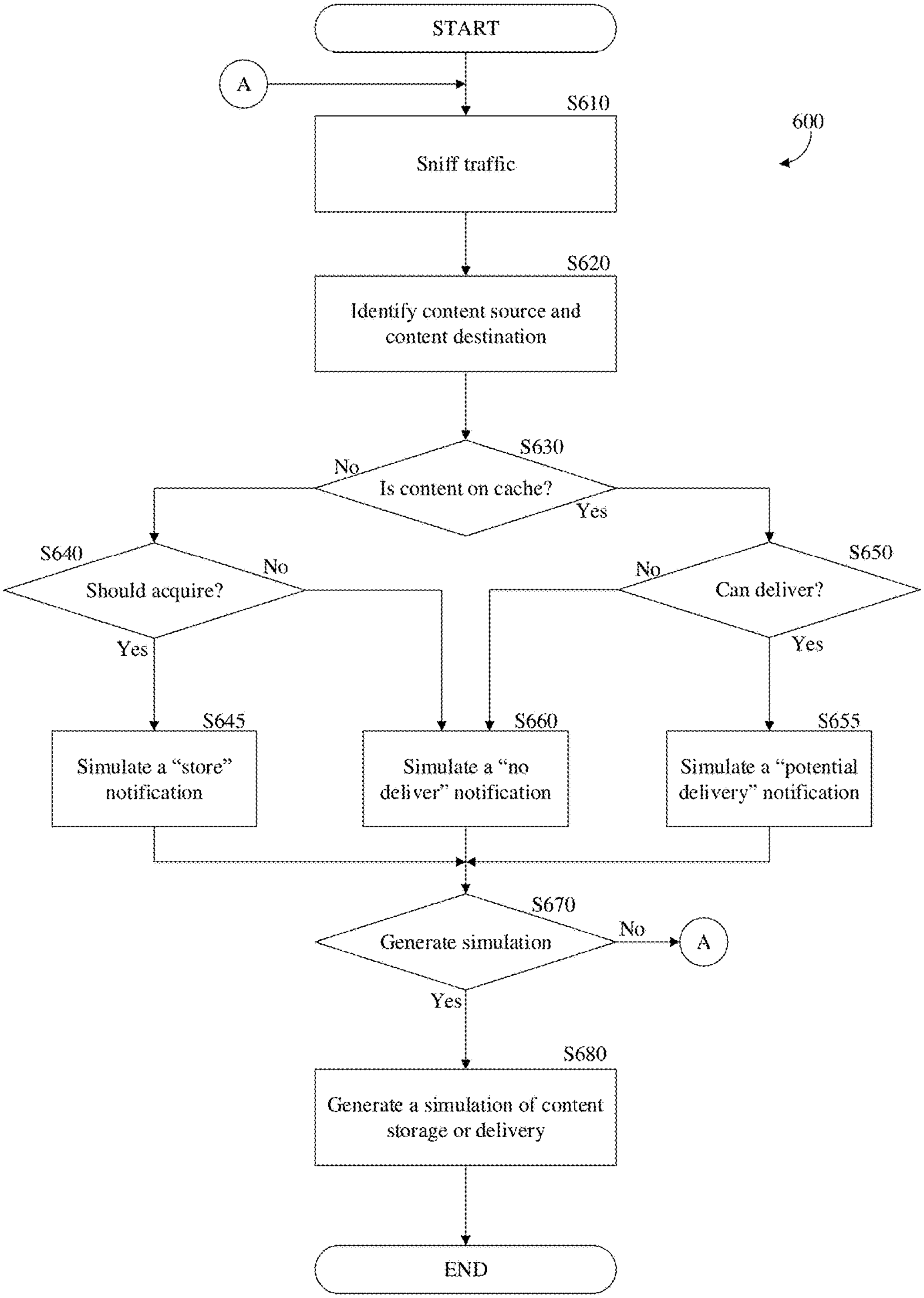

START
A
S610
Sniff traffic
600
S620
Identify content source and content destination
S630
Is content on cache?
No
Yes
S640
Should acquire?
No
Yes
S650
Can deliver?
No
Yes
S645
Simulate a "store" notification
S660
Simulate a "no deliver" notification
S655
Simulate a "potential delivery" notification
S670
Generate simulation
No
A
Yes
S680
Generate a simulation of content storage or delivery
END

FIGURE 6

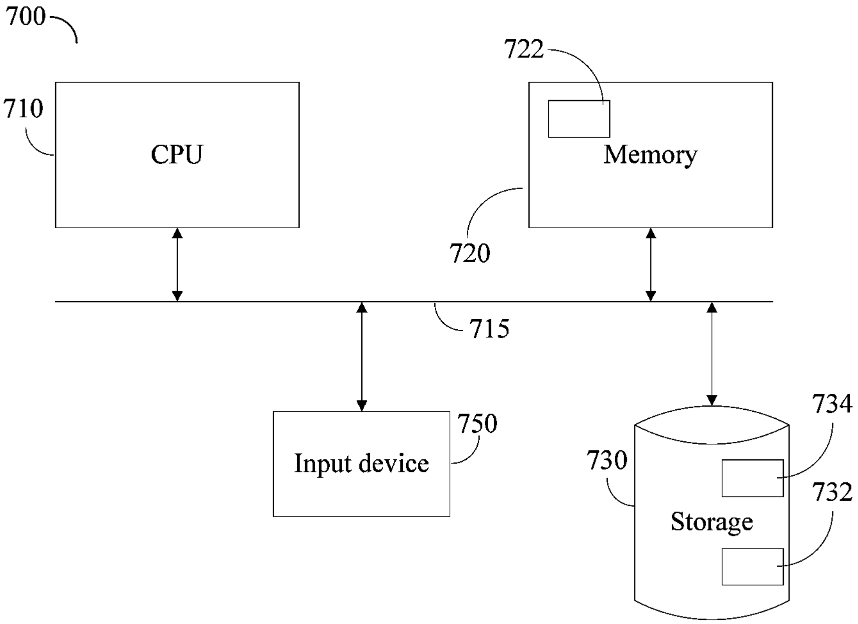
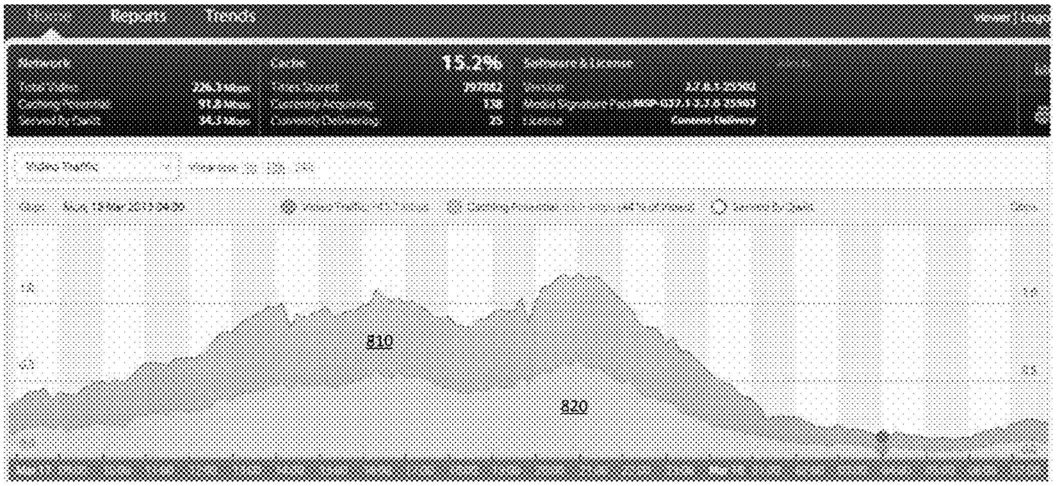
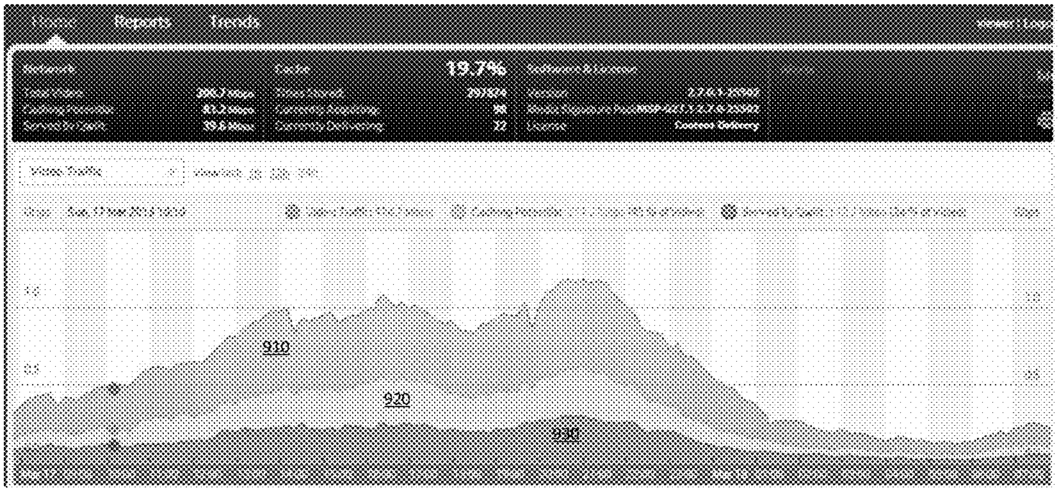

700
722
710
CPU
Memory
720
715
750
Input device
734
730
732
Storage

FIGURE 7



Network		Cache		Software & License	
Total Video	226.3 Mbps	Titles Stored	397882	Version	[illegible]
Caching Potential	91.8 Mbps	Currently Acquiring	138	Media Signature Pack	[illegible]
Served By QwiLt	34.3 Mbps	Currently Delivering	25	License	Content Delivery

FIGURE 8



Network		Cache		Software & License	
Total Video	206.7 Mbps	Titles Stored	397824	Version	[illegible]
Caching Potential	83.2 Mbps	Currently Acquiring	98	Media Signature Pack	[illegible]
Served By QwiLt	39.6 Mbps	Currently Delivering	22	License	Content Delivery

FIGURE 9

SYSTEM AND METHOD OF PERFORMING ANALYTICS WITH RESPECT TO CONTENT STORING SERVERS CACHING POPULAR CONTENT

CROSS-REFERENCE TO RELATED APPLICATION

This application claims priority from U.S. provisional patent application No. 61/969,418 filed Mar. 24, 2014 and is a continuation-in-part of U.S. patent application Ser. No. 13/006,785 filed Jan. 14, 2011, entitled “A System for Detection of Content Servers and Caching Popular Content Therein”, which claims priority from U.S. provisional patent application No. 61/375,836, entitled “A System for Detection of Content Servers and Caching Popular Content Therein”, filed on Aug. 22, 2010, all of the above-noted applications being assigned to a common assignee and all of the above-noted applications are hereby incorporated by reference in their entirety.

BACKGROUND

1. Technical Field

The present disclosure generally relates to classification of packets transmitted over a network and particularly to the determination of content to be stored in storage located along the network based on the type of data transferred in the packet. Even more particularly, the present disclosure relates to traffic analytics respective of the data and generation of a simulation respective of the analysis.

2. Description of the Related Art

Service providers worldwide are facing a mounting problem of flattening revenues coupled with increasing costs brought forth by increasing usage of bandwidth, growing churn, subscriber saturation and competitive price pressures. These mobile and broadband providers are facing challenges in delivering new revenue generating services while seeing their over the top (OTT) counterparts reaping rewards with content delivered over the networks they built and maintained over the years.

The vast majority of these OTT services are delivered over hypertext transfer protocol (HTTP), the de-facto protocol for application development and delivery. Be it video, social networking, search, or advertising, and be it over fixed line or mobile applications, these OTT services are most likely running on top of HTTP. However, this protocol is also the most processing intensive protocol for network devices. Hence, practically any increase in usage increases the strain on the service providers.

Certainly, one way to control traffic on the Internet requires various levels of understanding of the traffic that flows through the network, and that understanding is also increasing in its level of sophistication. Various systems and solutions have been offered in the related art that utilize deep-packet-inspection (DPI) to enable an ever sophisticated examination of network traffic and ability to shape the traffic on the network. This ability allows the service providers to better manage the network and its related resources and provide a higher level of quality of service (QoS) in the hopes of increasing revenues and profits.

SUMMARY

However, the rapid increase in the delivery of heavy bandwidth consuming data, such as video, and consumption thereof, requires a new level of handling that is not available today in related art solutions. A known problem is the access of a user to a content source and subsequently the access by another user to the same content, resulting in additional load on the content provider and on the entire network. When, for example, popular video clips are accessed, there is a significant and noticeable degradation of network performance which may even lead to a network failure. Some related art solutions attempt to store all the data in caches. However, the huge amounts of data and the need to inspect each and every packet, regardless of its source, makes this a daunting and impractical task.

It would be advantageous to provide service providers with a solution that would enable them to effectively manage and control the delivery of heavy bandwidth consuming data such that the overall bandwidth requirements are loaded and better shared across the network in general, and in particular within the network of a specific service provider. It would be further advantageous to provide a system and method for performing traffic analytics that determines the effect such a solution provides or could provide.

Accordingly, one or more exemplary embodiments provide a solution that effectively manages and controls the delivery of heavy bandwidth consuming data such that the overall bandwidth requirements are loaded and better shared across the network, particularly within the network of a specific service provider. Further, techniques are provided for performing traffic analytics that determine the effect such a solution provides or could provide.

The foregoing and/or other aspects of the exemplary embodiments may be achieved by a computerized method of generating a traffic simulation respective of at least one content storing server for caching popular content. This method may include sniffing traffic between at least a first portion of a network and at least a second portion of the network, identifying, from the sniffed traffic, at least a source of content and at least a destination of the content, determining if the content is stored on a cache, simulating a notification respective of the content, and generating a simulation of traffic respective at least of: the content and the simulated notification.

The computerized method may further include determining, upon determining that the content is not stored on the cache, if the content should be acquired to be stored on the cache.

The computerized method may further include simulating, upon determining that the content should be acquired, a “store” notification respective of the content.

The computerized method may further include simulating, upon determining that the content should not be acquired, a “no deliver” notification respective of the content.

The computerized method may further include determining, upon determining that the content is stored on the cache, if the content can be delivered from the cache.

The computerized method may further include simulating, upon determining that the content can be delivered from the cache, a “potential delivery” notification respective of the content.

The computerized method may further include simulating, upon determining that the content cannot be delivered from the cache, a “no deliver” notification respective of the content.

The identifying may be performed by a deep-packet-inspection (DPI) unit.

The computerized method may further include receiving a size of the cache for storing the content, wherein the generating may further include generating the simulation of traffic respective of the cache size.

The computerized method may further include determining, upon determining that the content is not stored on the cache, if the content should be acquired to be stored on the cache.

The computerized method may further include generating, upon determining that the content should be acquired, a "store" notification respective of the content.

The computerized method may further include generating, upon determining that the content should not be acquired, a "no deliver" notification respective of the content.

The computerized method may further include determining, upon determining that the content is stored on the cache, if the content can be delivered from the cache.

The computerized method may further include generating, upon determining that the content can be delivered from the cache, a "potential delivery" notification respective of the content.

The computerized method may further include generating, upon determining that the content cannot be delivered from the cache, a "no deliver" notification respective of the content.

The foregoing and/or exemplary embodiments may be achieved by an apparatus for generating a traffic simulation respective of at least one content storing server. The apparatus may include a first network interface to sniff traffic between at least a first portion of a network and at least a second portion of the network, a storage for storing at least information respective of content received through the first network interface, a second network interface configured to communicate with the at least first portion of the network and the at least second portion of the network, a processing unit; and a memory containing instructions that, when executed by the processing unit, cause the apparatus to: sniff traffic between at least the first portion of the network and at least the second portion of the network; identify from the sniffed traffic at least a source of the content and at least a destination of the content; determine if the content is stored on a cache; simulate a notification respective of the content; and generate a simulation of traffic respective at least of: the content and the simulated notification.

The apparatus may further include a deep-packet-inspection (DPI) unit coupled to the first network interface and configured to identify at least the source of the content respective of the sniffed traffic, the DPI unit further configured to inspect one or more packets provided, through the first network interface, from the identified source of the content, each packet having at least a specific source address and a specific destination address.

The memory may further contain instructions that, when executed by the processing unit, cause the apparatus to: determine, upon determining that the content is not stored on the cache, if the content should be acquired to be stored on the cache.

The memory further contains instructions that, when executed by the processing unit, cause the apparatus to: simulate, upon determining that the content should be acquired, a "store" notification respective of the content.

The memory further contains instructions that, when executed by the processing unit, cause the apparatus to: simulate, upon determining that the content should not be acquired, a "no deliver" notification respective of the content.

The memory further contains instructions that, when executed by the processing unit, cause the apparatus to:

determine, upon determining that the content is stored on the cache, if the content can be delivered from the cache.

The memory further contains instructions that, when executed by the processing unit, cause the apparatus to: simulate, upon determining that the content can be delivered from the cache, a "potential delivery" notification respective of the content.

The memory further contains instructions that, when executed by the processing unit, cause the apparatus to: simulate, upon determining that the content cannot be delivered from the cache, a "no deliver" notification respective of the content.

The foregoing and/or other aspects of the exemplary embodiments may be achieved with a non-transitory computer readable storage medium storing a program for executing a method of generating a traffic simulation respective of at least one content storing server, the content storing server operative for caching popular content, the method including sniffing traffic between at least a first portion of a network and at least a second portion of the network, identifying, from the sniffed traffic, at least a source of content and at least a destination of the content, determining if the content is stored on a cache; simulating a notification respective of the content, and generating a simulation of traffic respective at least of: the content and the simulated notification.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features, and advantages will become apparent and more readily appreciated from the following detailed description taken in conjunction with the accompanying drawings in which:

FIG. **1**A is a block diagram of a basic network system operating as a bump-in the wire according to an exemplary embodiment;

FIG. **1**B is a block diagram of a basic network system operating as a sniffer according to an exemplary embodiment;

FIG. **2** is a block diagram of an apparatus to identify content sources and provide content from the apparatus according to an exemplary embodiment;

FIG. **3** is a flowchart depicting the identification of a content source according to an exemplary embodiment;

FIG. **4** is a flowchart depicting the storage of content from identified content sources in the storage of the apparatus according to an exemplary embodiment;

FIG. **5** is a flowchart describing the providing of content to a requesting node according to an exemplary embodiment;

FIG. **6** is a flowchart of a method for performing analytics with respect to content storing servers caching popular content according to an exemplary embodiment;

FIG. **7** is a schematic illustration of a server implemented according to an exemplary embodiment;

FIG. **8** is a graph of network traffic for an exemplary network; and

FIG. **9** is another graph of network traffic for an exemplary network.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

Below, exemplary embodiments will be described in detail with reference to accompanying drawings so as to be easily realized by a person having ordinary knowledge in the art. The exemplary embodiments may be embodied in various forms without being limited to the exemplary embodiments set forth herein. Descriptions of well-known parts are omitted for clarity, and like reference numerals refer to like elements throughout.

It is important to note that the embodiments here disclosed are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claims. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality.

Disclosed are systems and methods for performing a simulation of efficient usage of a bandwidth of network transferring content, such as video data, when using caching techniques. Packets from content sources are analyzed to identify content sources that provide predetermined types of data. Upon identification of such content sources, any data that arrives from such a content source is subject to a deep-packet-inspection (DPI) process to positively identify the content and determine whether or not to store it in cache storage so that when a subsequent request for the same content is received, there is no need to transmit the content from the content source, since it can simply be delivered from the system's storage. The results include at least less traffic going on the entire network, faster service, and lower operational costs. The simulation provides a prediction of potential effectiveness of the system for a given network and its respective traffic when being used.

Reference is now made to FIG. **1**A where an exemplary and non-limiting block diagram **100**A of a basic network system according to an exemplary embodiment is shown. The system depicted comprises of a global network **120** and a service provider network **130** that are coupled by a 'bump-in-the-wire' apparatus **110**. While the network **120** and **130** are shown as detached from each other it should be noted that this is only an exemplary configuration and other configurations are possible without departing from the principles of the present disclosure. A network may be a Local Area Network (LAN), Wide Area Network (WAN), Metro Area Network (MAN), the World Wide Web (WWW), the Internet, combinations thereof, or the like. To the global network **120** there are connected one or more content sources (CSs), shown as CS_1 **140-1** through CS_n **140-*n***, commonly referred to as CS **140**. The content sources provide content, such as video clips, upon request from the appropriate CS to a user. To the service provider network **130** there are connected one or more user nodes (UNs), shown as UN_1 **150-1** through UN_m **150-*m***, commonly referred to as UN **150**.

When a UN **150** requests content from a CS **140** it is transferred, according to the exemplary embodiment, through a service delivery apparatus (SDA) **110**, the function of which is described in more detail herein below. Generally, the SDA **110** may provide the requested content from its storage or, when such content, or portions thereof, are not in the SDA **110**, then the request is forwarded to the appropriate CS **140** for the delivery of the content, as further described below.

FIG. **1**B depicts an exemplary and non-limiting block diagram **100**B of a basic network system operating as a sniffer according to an exemplary embodiment. Operating similarly to the system shown in FIG. **1**A, the system **100**B has the SDA **110** connected in a sniffer mode, i.e., the SDA merely listens to the traffic moving between the global network **120** and the service provider network **130** without routing packets through itself. When the SDA **110** is connected in sniffer mode it will typically connect to the service provider network **130** through a separate communication port to provide content stored therein to a destination connected to the service provider network **130**.

FIG. **2** provides an exemplary and non-limiting block diagram of the SDA **110** that identifies content sources and provides content from the apparatus in accordance with the principles of the present disclosure. The SDA **110** comprises a DPI unit **210**, a storage **220**, a content delivery unit (CDU) **230** and an input/output interface **240**. According to the principles of the present disclosure, the DPI unit **210** has at least two separate tasks. The first task is to identify sources of content that potentially contain data that may be worthwhile to store in storage **220**. For example, video servers may be located throughout the global network **120** and randomly accessed by UNs **150** of the service provider network **130**. In order to overcome the deficiencies of related art solutions, the SDA **110** is implemented differently.

The DPI unit **210** is provided with data types to look for in data packets that are transmitted through the SDA **110**. Instead of inspecting each and every packet, the DPI unit **210** may inspect only one-in-a-thousand packets out of the entire traffic thereby significantly lowering the processing load. It should be understood that the method for selecting the sampled packets is typically not performed by using a simple counter to process one out of every predetermined number of packets. Instead, the source and destination addresses from each packet are fed into a hash function, and the hash function result is compared to a configurable threshold, and the result of this comparison determines if the packet is inspected or not. In addition, it should be understood that the hash function is symmetric with respect to the source and destination addresses, such that swapping the source address and the destination address does not change the hash result.

In one exemplary embodiment, the source and destination ports may also be used as part of the hash function operation. This is needed to guarantee that each flow comprising of multiple packets sent between a UN **150** and a CS **140** is either fully ignored or fully inspected.

Upon determination that a specific CS **140** provides a desired data type, the identification of that CS **140** is stored. Any future packet received from or sent to the identified CS **140** is inspected by the DPI unit **210** and if the packet contains content that may be interesting for storage, such as video content, such content is stored in the storage **220**.

This kind of inspection ensures that demand for highly popular content from a popular CS **140** is likely to be quickly detected while infrequent access to a CS **140** would typically not impact the traditional operation of the system. It should be noted that identification of a CS **140** does not have to be on the first detection of data of interest and threshold levels may be used, as well as an aging mechanism so that relatively infrequently accessed CSs **140** would lose older accesses, and those older accesses would therefore not accumulate to reach a threshold value. The threshold may be static, dynamic or adaptive. Static thresholds are predetermined thresholds that remain constant. Dynamic thresholds are forcefully changed, for example, at a certain time of day, or according to information respective of the user. Adaptive thresholds are changed in response to changes in characteristics of the website engagement and may vary depending on a variety of parameters.

While DPI unit **210** operates on the packets that arrive from CSs **140**, the CDU **230** operates with respect of requests for content received from the UNs **150** of the service provider network **130**. Upon receipt of such a request, the DPI **210** first checks if content from the requested CS **140** actually resides in the storage **220** by first checking that the CS **140** identification is known to the SDA **110**. If that is the case, then the storage **220** is checked for the possibility of delivery of the content or portions thereof. If the entire content or portions thereof are found, then these are delivered to the requesting UN **150**. If the entire content is missing, or certain portions thereof are missing, then the request is forwarded to the appropriate CS **140**. Storage **220** may be semiconductor media, magnetic media, or any other type of storage media appropriate for the storage of content.

Reference is now made to FIG. **3** that depicts an exemplary and non-limiting flowchart **300** depicting the identification of a content source in accordance with an exemplary embodiment. In S**310** there are received and/or fetched parameters relating to the data of interest in the CSs. For example, the parameters may include parameters pertaining to video data. In S**320**, packets are selected off of the network traffic, for example the global network **120**. The ratio between the number of packets that pass through the network and the number inspected may be configured, so it could be one-in-a-thousand, one-in-ten-thousand, and so on and so forth.

In S**330**, it is checked if the data in the packet corresponds to the data parameters, e.g., contains video data, and if so, execution continues with S**340**; otherwise, execution continues with S**370**. In S**340**, the count respect of the CS **140** that is the source of the packet is updated, for example but not by way of limitation, by incrementing the value of a counter. In S**350**, it is checked if the count for that CS **140** has exceeded a threshold value and if so execution continues with S**360**; otherwise, execution continues with S**370**. In one exemplary embodiment, the count may also have an aging mechanism as described previously (not shown). Furthermore, different data types may have different thresholds, different count increases, and different count aging. In S**360**, the CS **140** is identified as a source of content eligible for storage in a storage unit, for example, storage **220**. In S**370**, it is checked if there are more data packets to be inspected and if so execution continues with S**320**; otherwise execution terminates.

Reference is now made to FIG. **4** that depicts an exemplary and non-limiting flowchart **400** depicting the storage of content, from the identified CS **140**, in the storage **220** of the SDA **110** according to the exemplary embodiment. In S**410** a packet is received by SDA **110**. In S**420**, it is checked whether the received packet is from an identified CS **140** and if so execution continues with S**430**; otherwise execution continues with S**460**. In S**430**, the received packet is inspected by the DPI unit **210** to possibly identify content of interest. It should be understood that this takes place as it is possible that even though the packet arrived from an identified CS **140** it does not contain content of interest and therefore there is no need to waste valuable storage space in storage **220** for that data.

In S**440**, it is checked whether such content of interest was found and if so execution continues with S**450**; otherwise, execution continues with S**460**. In S**450**, the content from the received packet is stored in storage, for example, storage **220**. In S**460**, it is checked whether more packets are received and if so execution continues with S**410**; otherwise, execution terminates.

Reference is now made to FIG. **5** that depicts an exemplary and non-limiting flowchart **500** describing the providing of content to a requesting UN **150** according to the principles of the exemplary embodiment. In S**510**, the SDA **110** receives a request for content from a UN **150**. In S**520**, it is checked if the requested content is in an identified CS **140** and if so execution continues with S**530**; otherwise, execution continues with S**560**.

In S**530**, it is checked whether the requested content is in storage, for example storage **220**, and if so execution continues with S**540**; otherwise, execution continues with S**560**. In S**540**, it is checked whether the whole content is in storage and if so execution continues from S**550**; otherwise, execution continues with S**560**. The need to check storage twice (i.e., in operations S**530** and S**540**) arises because the requested content is not the same as the whole content. In a video file, for example, the file is broken up into smaller pieces. In this exemplary embodiment, the requested content would be one of the smaller pieces, while the whole content would be the entire video file.

In S**550**, the content is delivered to the requesting UN **150**. In S**560**, it is checked whether additional content requests exist and if so execution continues with S**510**; and if not, execution terminates.

In one exemplary embodiment, when detecting that a portion of the requested content is in the storage **220** and deliverable to the requesting UN **150**, such content is delivered immediately to the UN **150** while only the missing portions of the content is requested from the CS **140**. Hence a request from the CDU **230** may be for the requested content or portions thereof. It should be further understood that in an exemplary embodiment, once the DPI unit **210** determines that a CS **140** may contain content that should be stored in storage **220**, the packets from such a CS **140** are consistently inspected for determination of popular content.

FIG. **6** is a non-limiting exemplary flowchart **600** of a method for performing analytics with respect to content storing servers caching popular content according to an exemplary embodiment. In S**610**, sniffing traffic between at least a first portion of a network and at least a second portion of a network is performed in order to identify content. FIG. **1**B is a non-limiting example for a basic network system operating as a sniffer. Sniffing may be performed, for example, by SDA **110**.

In S**620** a content source and a content destination are identified, respective of the sniffed traffic. Further information may be identified, such as type of content, size of content, time stamps and the like. In S**630** a check is performed to determine if the identified content is in a cache of an apparatus, such as SDA **110**. If the content is in a cache, execution continues at S**650**, otherwise execution continues at S**640**.

In S**640**, a check is performed to determine if the content should be acquired. If the content should be acquired, execution continues at S**645**, otherwise execution continues at S**660**. In S**645**, a "store" notification is simulated, indicating that the identified content should be stored in a cache. The notification may be stored in a memory and used to generate a simulation.

In S**650**, a check is performed to determine if the identified content can be delivered from the cache. An example of when content might be stored in the cache but cannot be delivered would be if there are copyright issues. For instance, if the content stored in the cache is licensed for viewing only within one country, the content cannot be delivered if the UN **150** is in another country. If the content can be delivered from the cache, execution continues at S**655**, otherwise execution continues at S**660**. In S**655**, a "potential delivery" notification is simulated, indicating that content stored in a cache may be delivered. The notification may be stored in a memory and used to generate a simulation.

In S**660**, a "no deliver" notification is simulated, indicating that the identified content cannot be delivered. The notification may be stored in a memory and used to generate a simulation. In certain exemplary embodiments, a further check may be performed to determine if content which cannot be delivered should be deleted from the cache.

A check is performed in S**670** to determine if the results of the simulation should be generated. For example, it may not be desirable to generate a simulation if a sufficient amount of data points have not yet been collected. If it is determined that the results of the simulation should be generated, execution continues at S**680**, otherwise execution continues at S**610**. In S**680**, the results of the simulation are generated, respective of at least one simulated notification of any of S**645**, S**655** or S**660**, perhaps in the form of a graph as shown in FIG. **9**. The results of the generated simulation may be stored in a memory.

In certain embodiments, actual notifications respective of S**645**, S**655**, and S**660**, or combination of actual and simulated notifications may be used to generate the results of the simulation. An actual notification is generated by the SDA **110** during its course of run. A simulated notification is generated during a "what-if" scenario. For example, if a lower threshold by which acquisition decisions are made is simulated (to acquire more content), additional notifications would be created.

In an embodiment, the results of the simulation may be an illustration of the network operating with the SDA **110** described in more detail herein, and may also include a comparison to the same network operating without the SDA **110** present. The results of the simulation may be generated in real-time or at a later time using notifications stored as information in a memory. The results of the simulation may further be respective of limitations of the SDA **110**, such as the physical size of the cache. In such an exemplary embodiment, the results of the simulation may be generated by simulating any cache size to determine the content which would be available for delivery, and actual notifications would correspond to the actual cache size while simulated notifications would correspond to the simulated cache size.

This allows for more efficient use of resources, since a larger (and therefore costlier) cache size would not necessarily result in more content being delivered. By simulating different cache sizes, it is possible to find an optimal cache size for the content storing server. The method shown herein may be performed by the SDA **110** itself, being appropriately configured, or otherwise, by a simulator device having access to data from the SDA **110** via its network connectivity interfaces.

FIG. **7** depicts an exemplary and non-limiting schematic illustration of server **700**, implemented according to an exemplary embodiment. In one embodiment the server **700** is communicatively connected to the SDA **110** (not shown) for collecting the information therefrom. In an exemplary embodiment, the server **700** is connected like the sniffer mode of the SDA **110** as shown in FIG. 1B, for the purpose of collecting the desired information. The server **700** comprises at least a processing element **710**, for example, a central processing unit (CPU) that is coupled via a bus **715** to a memory **720**. The memory **720** further comprises a memory portion **722** that contains instructions that when executed by the processing element **710** performs the method described in more detail herein. The memory may be further used as a working scratch pad for the processing element **710**, a temporary storage, and others, as the case may be. The memory may comprise of volatile memory such as, but not limited to, random access memory (RAM), or non-volatile memory (NVM), such as, but not limited to, flash memory.

The processing element **710** may be coupled to an input device **750**, e.g., a mouse and/or a keyboard, and a data storage **730**. Data storage **730** may be used for the purpose of holding a copy of the method executed according to the disclosed technique in FIG. **6**. Data storage **730** may further comprise storage portion **732** containing information respective of at least a content packet. The information may include source, destination, packet size and the like. Data storage **430** may further comprise storage portion **434** containing at least the results of the simulation generated respective of the information.

Reference is now made to FIGS. **8** and **9**, which are non-limiting exemplary graphs of network traffic for exemplary networks.

FIG. **8** is a plot of actual network traffic in Gigabits per second (Gbps) as a function of time. A first graph area **810** indicates the total amount of video traffic for the given network. A second graph area **820** indicates the total amount of video traffic determined by a system, for example server **700**, to be potentially stored and delivered by a cache.

FIG. **9** is another plot of network traffic in Gigabits per second (Gbps) as a function of time. A first graph area **910** indicates the total amount of video traffic for the given network. A third graph area **930** indicates the amount of data which is delivered by a cache. The first graph area **910** and the third graph area **930** together comprise information. The information is used, for example by a server **700** implementing the method of FIG. **6** to generate a simulation of a network operating with the apparatus described in more detail herein. The results of the generated simulation are represented by a second graph area **920**, which indicates the amount of additional traffic which may be stored and delivered by a cache.

The relationship between FIGS. **8** and **9** can be explained as follows. Graph area **810** corresponds to area **910**, and area **820** corresponds to area **920**. Area **810** is the total amount of video traffic which the SDA has access to. Area **820** is the total amount of video which is eligible for caching (according to thresholds, etc.). Area **930** is what can be delivered by the SDA. For example, in FIG. **9**, if the cache size is increased, more content would not necessarily be delivered from the cache, as what is delivered has a smaller area than what is cache-eligible.

The principles of the disclosure are implemented as hardware, firmware, software or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPUs"), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit and/or display unit.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the present disclosure and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

What is claimed is:

1. A network traffic simulation tool for at least one content storing server, comprising:

a first network interface to sniff traffic between at least a first portion of a network and at least a second portion of the network;

a storage for storing at least information for content received through the first network interface;

a second network interface configured to communicate with the at least first portion of the network and the at least second portion of the network;

a processing unit; and

a memory containing instructions that, when executed by the processing unit, cause the apparatus to:

sniff traffic between at least the first portion of the network and at least the second portion of the network, wherein the sniffing listens to the traffic moving between the at least the first portion of the network and the at least the second portion of the network while the sniffed traffic passes directly between the at least the first portion of the network and the at least the second portion of the traffic;

identify from the sniffed traffic at least a source of the content and at least a destination of the content;

when the content is stored in a cache:

simulate a notification for the content; and

generate a simulation of traffic for at least: the content and the simulated notification.

2. The network traffic simulation tool of claim **1**, further comprising:

a deep-packet-inspection (DPI) unit coupled to the first network interface and configured to identify at least the source of the content for the sniffed traffic, the DPI unit further configured to inspect one or more packets provided, through the first network interface, from the identified source of the content, each packet having at least a specific source address and a specific destination address.

3. The network traffic simulation tool of claim **1**, wherein the memory further contains instructions that, when executed by the processing unit, cause the apparatus to: determine, upon determining that the content is not stored in the cache, if the content should be stored on the cache.

4. The network traffic simulation tool of claim **3**, wherein the memory further contains instructions that, when executed by the processing unit, cause the apparatus to: simulate, upon determining that the content should be stored, a "store" notification for the content.

5. The network traffic simulation tool of claim **3**, wherein the memory further contains instructions that, when executed by the processing unit, cause the apparatus to: simulate, upon determining that the content should not be acquired, a "no deliver" notification for the content.

6. The network traffic simulation tool of claim **1**, wherein the memory further contains instructions that, when executed by the processing unit, cause the apparatus to: determine, upon determining that the content is stored in the cache, if the content can be delivered from the cache.

7. The network traffic simulation tool of claim **6**, wherein the memory further contains instructions that, when executed by the processing unit, cause the apparatus to: simulate, upon determining that the content can be delivered from the cache, a "potential delivery" notification for the content.

8. The network traffic simulation tool of claim **6**, wherein the memory further contains instructions that, when executed by the processing unit, cause the apparatus to: simulate, upon determining that the content cannot be delivered from the cache, a "no deliver" notification for the content.

* * * * *

EXHIBIT K

US010673947B2

(12) **United States Patent**
Romem et al.

(10) **Patent No.: US 10,673,947 B2**
(45) **Date of Patent: Jun. 2, 2020**

(54) **SYSTEM AND METHOD FOR PROVIDING A CLIENT DEVICE SEAMLESS ACCESS TO A PLURALITY OF REMOTE STORAGE DEVICES PRESENTED AS A VIRTUAL DEVICE**

(71) Applicant: **Excelero Storage Ltd.**, Tel Aviv (IL)

(72) Inventors: **Yaniv Romem**, Jerusalem (IL); **Omri Mann**, Jerusalem (IL); **Ofer Oshri**, Kfar Saba (IL)

(73) Assignee: **Excelero Storage Ltd.**, Tel Aviv (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **16/282,629**

(22) Filed: **Feb. 22, 2019**

(65) **Prior Publication Data**

US 2019/0190989 A1 Jun. 20, 2019

Related U.S. Application Data

(63) Continuation of application No. 14/934,830, filed on Nov. 6, 2015, now Pat. No. 10,237,347.

(60) Provisional application No. 62/172,265, filed on Jun. 8, 2015.

(51) **Int. Cl.**

H04L 29/08	(2006.01)
G06F 16/30	(2019.01)
G06F 16/00	(2019.01)
G06F 3/06	(2006.01)
G06F 12/109	(2016.01)

(52) **U.S. Cl.**

CPC ***H04L 67/1097*** (2013.01); ***G06F 3/061*** (2013.01); ***G06F 3/067*** (2013.01); ***G06F 3/0665*** (2013.01); ***G06F 12/109*** (2013.01); ***G06F 16/00*** (2019.01); ***G06F 16/30*** (2019.01); *G06F 2212/1024* (2013.01); *G06F 2212/154* (2013.01); *G06F 2212/163* (2013.01); *G06F 2212/657* (2013.01)

(58) **Field of Classification Search**

None

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,745,671	A	4/1998	Hodges	
5,889,934	A	3/1999	Peterson	
6,108,812	A	8/2000	Born	
8,307,271	B1	11/2012	Liu et al.	
8,407,448	B1 *	3/2013	Hayden	G06F 9/45533 711/203
8,910,031	B1	12/2014	Liu et al.	

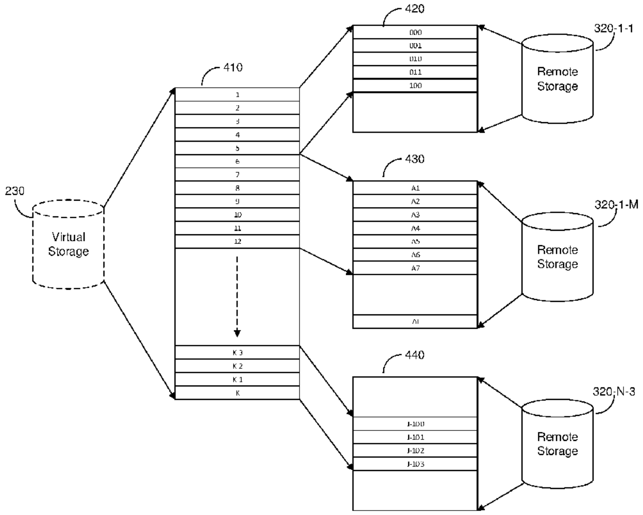
(Continued)

Primary Examiner — Philip J Chea
Assistant Examiner — Hassan A Khan
(74) *Attorney, Agent, or Firm* — M&B IP Analysts, LLC

(57) **ABSTRACT**

A computerized method for enabling a client device seamless access to a plurality of remote storage devices connected to the client device via a communication network. The method comprises receiving a plurality of physical addresses by a controller communicatively coupled to the client device and to a plurality of storage servers, each of the plurality of storage servers communicatively coupled to at least one storage device, the plurality of physical addresses enabling access by the controller to the remote storage devices. A single virtual storage device having a logical address space is generated on the device, wherein each of the plurality of physical addresses is mapped by the controller to a unique logical address of the virtual storage device.

11 Claims, 6 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

9,467,511	B2	10/2016	Tamir et al.	
9,467,512	B2 *	10/2016	Tamir	G06F 15/167
9,529,773	B2	12/2016	Hussain et al.	
9,639,457	B1	5/2017	Piszczek et al.	
2005/0038850	A1	2/2005	Oe et al.	
2006/0059408	A1	3/2006	Chikusa et al.	
2006/0235999	A1	10/2006	Shah et al.	
2008/0109616	A1	5/2008	Taylor	
2009/0300023	A1	12/2009	Vaghani	
2012/0079143	A1 *	3/2012	Krishnamurthi	H04L 49/90 710/39
2013/0073821	A1 *	3/2013	Flynn	G06F 3/061 711/162
2014/0211808	A1	7/2014	Koren et al.	
2015/0006663	A1 *	1/2015	Huang	G06F 11/3027 709/213
2015/0089121	A1 *	3/2015	Coudhury	G06F 12/0246 711/103
2015/0319237	A1 *	11/2015	Hussain	G06F 3/0605 709/217
2016/0034418	A1	2/2016	Romem et al.	
2016/0036913	A1	2/2016	Romem et al.	
2016/0057224	A1 *	2/2016	Ori	G06F 3/06 709/213
2016/0253267	A1 *	9/2016	Wood	G06F 16/00 711/117
2016/0266965	A1	9/2016	B et al.	

* cited by examiner

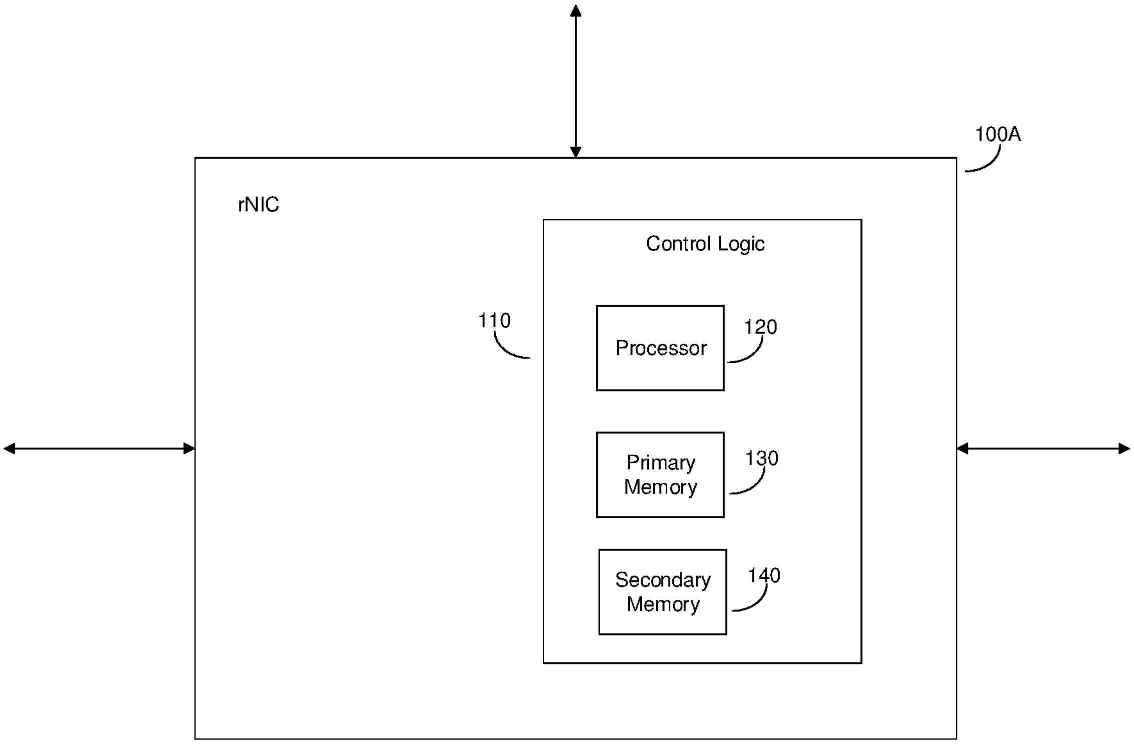

100A
rNIC
Control Logic
110
Processor
120
Primary Memory
130
Secondary Memory
140

FIGURE 1A

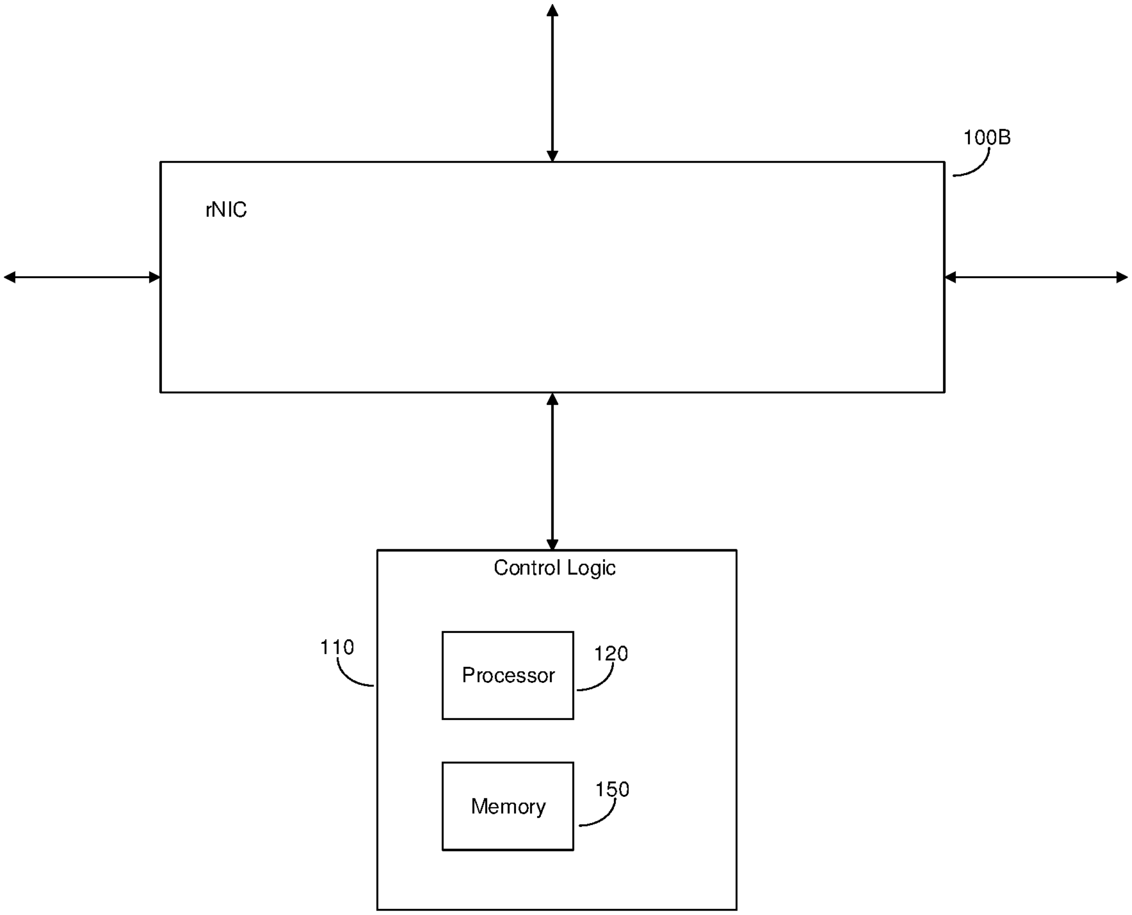


FIGURE 1B

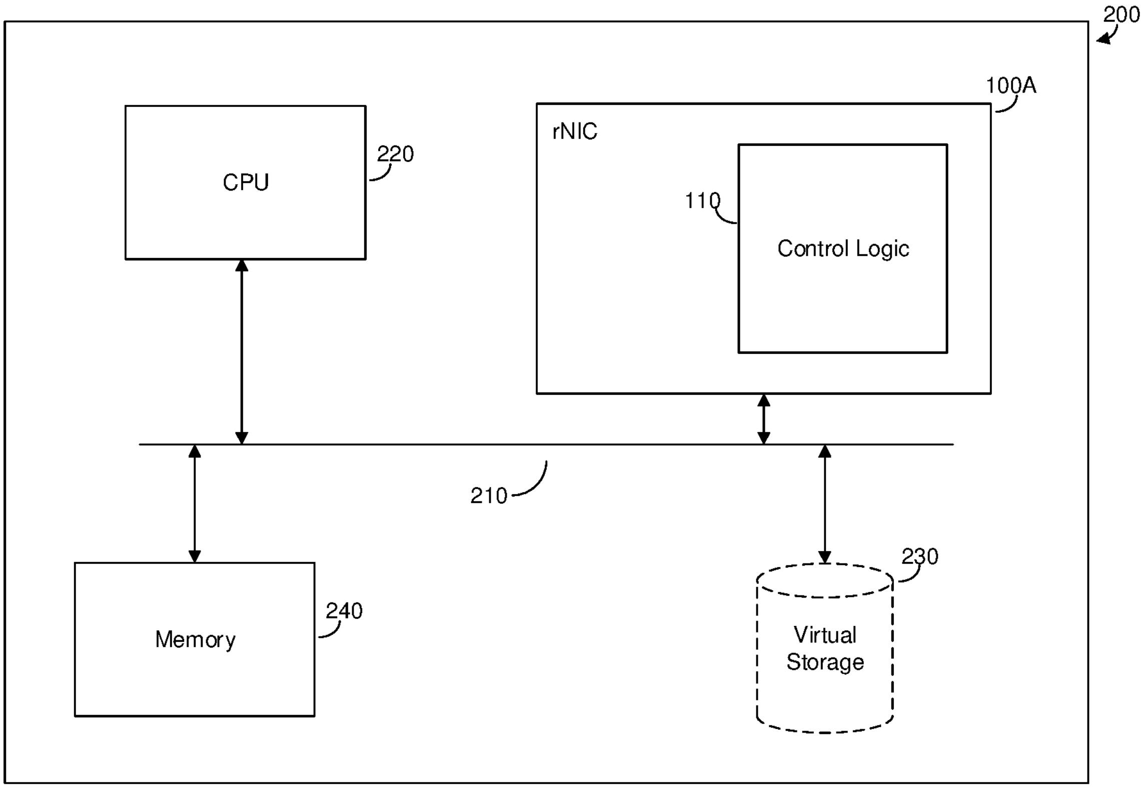

200
100A
rNIC
CPU
220
110
Control Logic
210
230
Virtual Storage
Memory
240

FIGURE 2

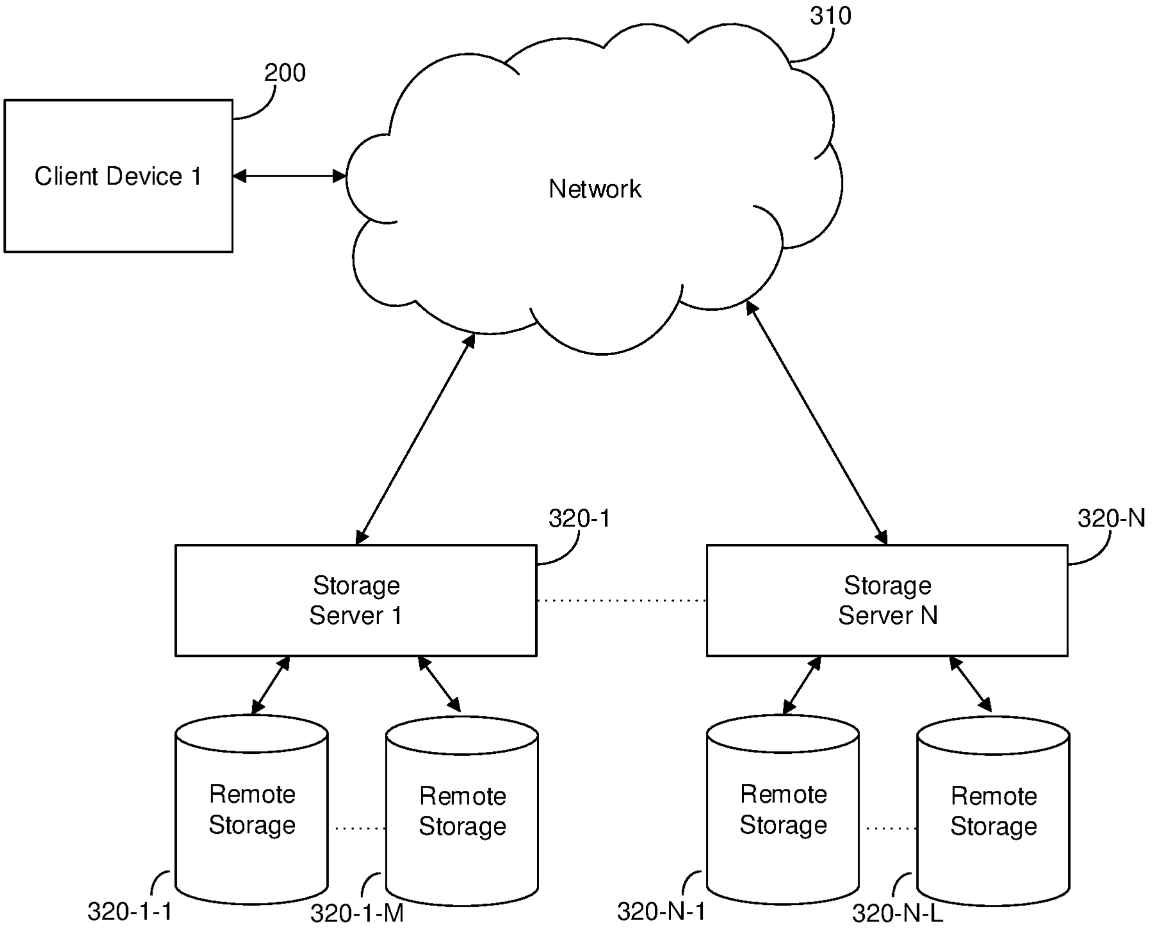

310
200
Client Device 1
Network
320-1
320-N
Storage Server 1
Storage Server N
Remote Storage
Remote Storage
Remote Storage
Remote Storage
320-1-1
320-1-M
320-N-1
320-N-L

FIGURE 3

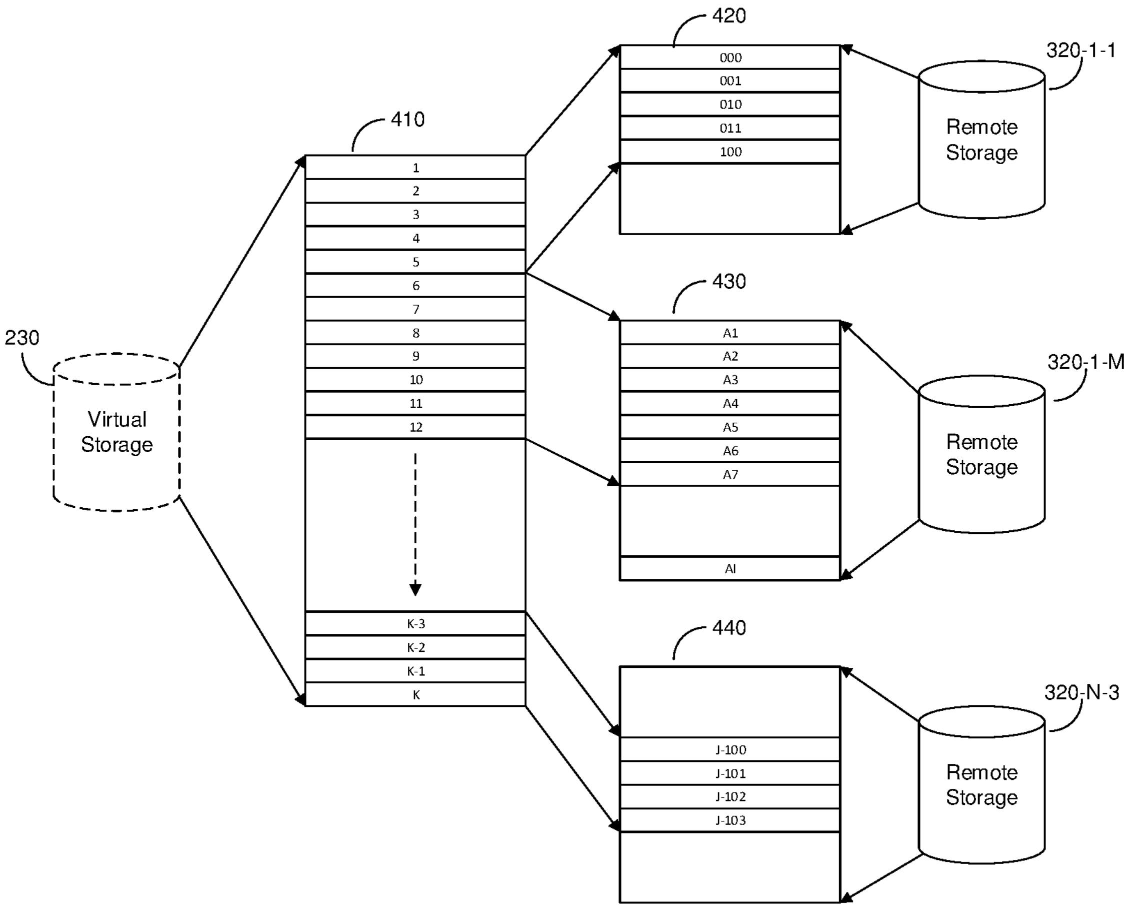

420
320-1-1
000
001
010
011
100
Remote Storage
410
1
2
3
4
5
6
7
8
9
10
11
12
K-3
K-2
K-1
K
430
A1
A2
A3
A4
A5
A6
A7
AI
320-1-M
Remote Storage
230
Virtual Storage
440
J-100
J-101
J-102
J-103
320-N-3
Remote Storage

FIGURE 4

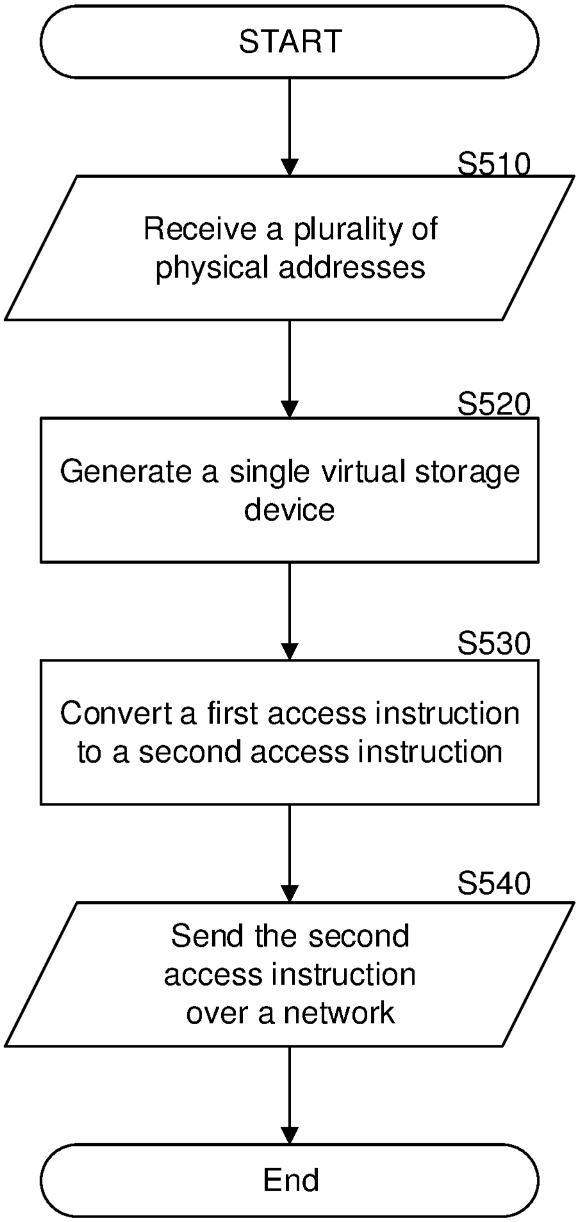

START
S510
Receive a plurality of physical addresses
S520
Generate a single virtual storage device
S530
Convert a first access instruction to a second access instruction
S540
Send the second access instruction over a network
End

FIGURE 5

SYSTEM AND METHOD FOR PROVIDING A CLIENT DEVICE SEAMLESS ACCESS TO A PLURALITY OF REMOTE STORAGE DEVICES PRESENTED AS A VIRTUAL DEVICE

This Application is a continuation of U.S. patent application Ser. No. 14/934,830, filed Nov. 6, 2015, now allowed, which claims the benefit of U.S. Provisional Application No. 62/172,265, filed on Jun. 8, 2015, the contents of which are incorporated herein by reference.

BACKGROUND

Field

The disclosure generally relates to remote storage access and particularly to remote storage access through use of a virtual storage device

Description of Related Art

The approaches described in this section are approaches that could be pursued, but not necessarily approaches that have been previously conceived or pursued. Therefore, unless otherwise indicated, it should not be assumed that any of the approaches described in this section qualify as prior art merely by virtue of their inclusion in this section. Similarly, issues identified with respect to one or more approaches should not assume to have been recognized in any prior art on the basis of this section, unless otherwise indicated.

Using remote direct memory access (RDMA) allows a client device to access remote storage devices. While this is advantageous in itself, approaches implemented to date use significant processing resources of the client device's central processing unit (CPU) which result in a high access latency. Additionally, such solutions do not offer dynamic use of the remote storage devices, which would allow the client device to request storage as needed.

SUMMARY

To realize some of the advantages there is provided a computerized method for enabling a client device seamless access to a plurality of remote storage devices connected to the client device via a communication network. The method comprises receiving a plurality of physical addresses by a controller communicatively coupled to the client device and to a plurality of storage servers, each of the plurality of storage servers communicatively coupled to at least one storage device, the plurality of physical addresses enabling access by the controller to the remote storage devices. A single virtual storage device having a logical address space is generated on the device, wherein each of the plurality of physical addresses is mapped by the controller to a unique logical address of the virtual storage device.

Another aspect of the disclosed teachings is a network interface card (NIC) for providing a client device seamless access to a plurality of remote storage devices connected to the client device via a communication network comprising a control logic. The control logic further comprises at least one processor and a memory. The memory includes instructions to enable the NIC to receive a plurality of physical addresses, the plurality of physical addresses enabling access by the NIC to the remote storage devices. The memory includes instructions to further generate a single

virtual storage device having a logical address space, wherein each of the plurality of physical addresses is mapped by the controller to a unique logical address of the virtual storage device. The memory further includes a map for translating each of a plurality of virtual addresses to a unique physical address of a remote storage device. Yet another aspect of the disclosed teachings is a communication system for providing a client device seamless access to a plurality of remote storage devices connected to the client device via a communication network. The system comprises a network interface card (NIC) and a control logic. The control logic communicates with the NIC. The control logic further comprises at least one processor and a memory, The memory includes instructions to further generate a single virtual storage device having a logical address space, wherein each of the plurality of physical addresses is mapped by the controller to a unique logical address of the virtual storage device. The memory further includes a map for translating each of a plurality of virtual addresses to a unique physical address of a remote storage device.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages will become apparent and more readily appreciated from the following detailed description taken in conjunction with the accompanying drawings, in which:

FIG. **1**A is a schematic illustration of a network interface controller (NIC) exemplifying some of the disclosed teachings for providing a client device seamless access to a plurality of remote storage devices connected to the client device via a communication network, implemented according to an exemplary embodiment.

FIG. **1**B is a schematic illustration of a NIC exemplifying some of the disclosed teachings for providing a client device seamless access to a plurality of remote storage devices connected to the client device via a communication network, implemented according to another exemplary embodiment.

FIG. **2** is a schematic illustration of client device implemented according to aspects of the disclosed teachings exemplifying an embodiment.

FIG. **3** is a schematic illustration of the client device described with respect to the exemplary embodiment of FIG. **2** connected with storage servers over a network.

FIG. **4** is a schematic illustration of a map of a virtual storage device according to some aspects of the disclosed teachings in accordance with an exemplary embodiment.

FIG. **5**—is a flowchart of a method for providing a client device seamless access to a plurality of remote storage devices connected to the client device via a communication network in accordance with an embodiment.

DETAILED DESCRIPTION

Below, exemplary embodiments will be described in detail with reference to accompanying drawings so as to be easily realized by a person having ordinary knowledge in the art. The exemplary embodiments may be embodied in various forms without being limited to the exemplary embodiments set forth herein. Descriptions of well-known parts are omitted for clarity, and like reference numerals refer to like elements throughout.

It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claims. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality.

FIG. **1**A is an exemplary and non-limiting schematic illustration of a network interface controller (NIC) **100**A for providing a client device seamless access to a plurality of remote storage devices connected to the client device via a communication network, implemented according to an exemplary embodiment. The NIC **100**A includes a control logic **110**, which further includes at least one processor **120** and a memory. In this illustrated embodiment, the memory comprises a primary memory **130**, and a secondary memory **140**. Primary memory **130** includes instructions that when executed by the processor **120** performs a method described in more detail herein. Secondary memory **140** includes a map for translating each of a plurality of virtual addresses to a unique physical address of a remote storage device. In some embodiments each virtual address is translated to a unique physical address belonging to one of a plurality of remote storage devices. The NIC **100**A is further configured to provide connectivity between the client device and a network. See, e.g. FIG. **3**. The NIC **100**A may be further configured to provide remote direct memory access to remote storage devices connected to one or more storage servers. Control logic **110** may be implemented, for example, as a field-programmable gate array (FPGA), a complex programmable logic device (CPLD), an application-specific integrated circuit (ASIC), a system on a chip (SoC), etc.

FIG. **1**B is an exemplary and non-limiting schematic illustration of a network interface controller (NIC) **100**B for providing a client device seamless access to a plurality of remote storage devices connected to the client device via a communication network, implemented according to another exemplary embodiment. The NIC **100**B is communicatively connected to a control logic **110**. The control logic **110** includes at least one processor **120**, and a memory **150**. The memory **150** includes a primary memory portion and a secondary memory portion (neither shown). The primary memory portion includes instructions that when executed by the processor **120** performs a method described in more detail herein. The secondary portion of the memory **150** includes a map for translating each of a plurality of virtual addresses to a unique physical address of a remote storage device. Of course the control logic **110** that is external to the NIC **100**B as shown in FIG. **1**B can include the primary memory **130** and the secondary memory **140** described above with respect to FIG. **1**A. Similarly, the control logic **110** that is internal to the NIC **100**A as shown in FIG. **1**A can include the memory **150** described above with respect to FIG. **1**B. Referring back to FIG. **1**B, the NIC **100**B is further configured to provide connectivity between a client device to a network (neither shown). See, for example, FIG. **3**. The NIC **100**B is further configured to provide remote direct memory access to remotely access a storage portion on a storage device communicatively connected a storage server (not shown). See, for example, FIG. **3**. Control logic **110** may be implemented, for example, as a field-programmable gate array (FPGA), a complex programmable logic device (CPLD), an application-specific integrated circuit (ASIC), a system on a chip (SoC), etc.

In certain embodiments, the control logic **110** as described in FIG. **1**A or FIG. **1**B may provide a client device with additional functionalities. For example, the control logic **110** may include a high availability module and a data protection module. Such modules may be provided, for example, by implementing protocols for redundant array of independent disks (RAID). In standard RAID levels, this may include RAID-1, RAID-5 and RAID-6, and in nested RAID levels, this may include RAID-10. The control logic **110** may further provide a space reduction module. A space reduction module may provide thin provisioning, deduplication, compression, and the like.

FIG. **2** is an exemplary and non-limiting schematic illustration of client device **200** implemented according to an embodiment. The client device **200** includes at least one processing element **220**, for example, a central processing unit (CPU). In an embodiment, the processing element **220** may include, or be a component of, a larger processing unit implemented with one or more processors. The one or more processors may be implemented with any combination of general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), field programmable gate array (FPGAs), programmable logic devices (PLDs), controllers, state machines, gated logic, discrete hardware components, dedicated hardware finite state machines, or any other suitable entities that can perform calculations or other manipulations of information. The processing element **220** is coupled via a bus **210** to a memory **240**. The bus supports a first communication protocol and may be, for example, implemented as a Peripheral Component Interconnect Express (PCI Express), Non Volatile Memory Express (NVMe), NVMe-over-Fabrics and the like. The memory **240** may be used as a working scratch pad for the processing element **220**, a temporary storage, and others, as the case may be. The memory may be a volatile memory such as, but not limited to random access memory (RAM), or nonvolatile memory (NVM), such as, but not limited to, Flash memory. The processing element **220** is further coupled with a NIC **100**A. Of course the client device **200** could include NIC **100**B, but for simplicity this exemplary embodiment will be described with reference to NIC **100**A. The processing element **220** is further coupled via bus **210** to a virtual storage **230**. Virtual storage **230** includes a plurality of virtual addresses. Each virtual address is mapped by the control logic **110** to a unique physical address. The unique physical addresses may correspond to a plurality of storage devices, such that the virtual storage includes at least a first physical address on a first storage device, and a second physical address on a second storage device. The storage devices may be communicatively coupled to one or more storage servers. A portion of the plurality of physical addresses may be physical addresses of a local storage device (not shown) communicatively coupled to the client. The virtual storage **230** is presented to the client device **200** by the control logic **110** as a local storage device and the client device **200** communicates to the virtual storage **230** using the first communication protocol. Access instructions are received by the control logic **110** for the virtual storage **230**. An access instruction includes at least one virtual address and an action to be performed corresponding to that virtual address. An action may be, for example, 'read', 'write', 'erase', etc. The access instruction is delivered over the first communication protocol and converted by the control logic **110** to the mapped physical address. The NIC **100**A sends the converted access instruction through a network over a second communication protocol to the storage device corresponding to the physical address. The second communication protocol may implement, in some embodiments, remote direct memory access (RDMA) protocols, such as but not limited to RDMA over Converged Ethernet (RoCE), Infiniband, and iWARP. In another exemplary embodiment, the communication may be over a peer-to-peer (P2P) network. The processing element **220** and/or the memory **240** may also include machine-readable media for storing software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the one or more processors, cause the processing system to perform the various functions described in further detail herein.

In some embodiments, the control logic **110** may associate the physical addresses with tiers, wherein the tier level reflects an access parameter, such as access latency. A threshold may be selected to determine if a physical address belongs to a first tier, a second tier, and so on. The first tier includes physical addresses having an access latency higher than physical addresses of the second tier. In a tiered system with more than two tiers, each tier includes physical addresses of storage devices which have an access latency that is within the threshold of that tier. Tiering of physical addresses may be advantageous, for example, in an exemplary embodiment where cloning of the data on the virtual storage device is performed, the cloned data may be stored with respect to physical addresses of the second tier, or below. Thus, data which is frequently used by the client device is available in low access latency, whereas replicated data which is used only in certain cases does not use resources with low access latency, thereby increasing the overall system efficiency.

FIG. **3** is a non-limiting exemplary schematic illustration of the client device **200** described with respect to FIG. **2** connected with storage servers **320-1** to **320**-N over a network **310**. Client device **200** is communicatively coupled with network **310**. The network **310** is configured to provide connectivity of various sorts, as may be necessary, including but not limited to, wired and/or wireless connectivity, including, for example, local area network (LAN), wide area network (WAN), metro area network (MAN), World Wide Web (WWW), Internet, and any combination thereof, as well as cellular connectivity. Network **310** is further communicatively coupled with storage servers **320**, such as storage servers **320-1** through **320**-N. Each storage server **320** includes one or more storage devices, such that storage server **320-1** provides access to remote storage devices **320-1-1** through **320-1**-M, or portions thereof, and storage server **320**-N provides access to remote storage devices **320**-N–1 through **320**-N-L, or portions thereof.

FIG. **4** is a non-limiting exemplary schematic illustration of a map of a virtual storage device **230** in accordance with an embodiment. Virtual storage device **230** includes a plurality of virtual addresses **410**. Each virtual address **1** through K is mapped to a unique physical address of a remote storage device. In this exemplary embodiment, virtual addresses 1 through 5 are mapped to physical addresses 000 through 100. Physical addresses 000 through 100 are a portion of the physical addresses **420** of remote storage device **320-1-1**. Further in this exemplary embodiment, virtual addresses 6 through 12 are mapped to physical addresses A1 through A7. Physical addresses A1 through A7 are a portion of the physical addresses **430** of remote storage device **320-1**-M. Virtual addresses K-3 through K are mapped to physical addresses J-100 through J-103. Physical addresses J-100 through J-103 are a portion of the physical addresses **440** of remote storage device **320**-N–3. The client device **200** communicates with the virtual storage device **230** as if it were a seamless local storage device. In certain embodiments, the control logic **100** may increase the size of the virtual storage device **230**, for example by having a first virtual address point to a plurality of secondary virtual addresses. The plurality of secondary virtual addresses are then each mapped to a unique physical address.

FIG. **5** is a non-limiting exemplary flowchart of a method for providing a client device **200** seamless access to a plurality of remote storage devices connected to the client device via a communication network in accordance with an embodiment. In S**510** a plurality of physical addresses are received by a control logic **110**. The control logic **110** is communicatively coupled to a client device **200**. The client device includes a NIC which provides connectivity to a plurality of storage servers, such as storage servers **320-1** through **320**-N. Each of the plurality of storage servers is communicatively coupled to at least one storage device. The plurality of physical addresses enable the control logic **110** to access the remote storage devices. In S**520** a single virtual storage device having a logical address space is generated on the client device **200**, wherein each of the plurality of physical addresses is mapped by the control logic to a unique logical address of the virtual storage device. The single virtual storage device may be generated using input/output (I/O) Virtualization. The I/O Virtualization may be implemented, for example, as Single Root I/O Virtualization (SR-IOV). In some embodiment, SR-IOV may be used to generate a plurality of single virtual storage devices, each implemented in accordance with the methods described herein. In S**530** a first access instruction is converted by the control logic **110**. An access instruction includes at least one virtual address and an action to be performed by the storage device corresponding to that virtual address. An action may be, for example, 'read', 'write', 'erase', etc. The first access instruction is received over a first communication protocol. In an embodiment, the first communication protocol is NVMExpress. The first access instruction includes at least a first address within the logical address space. The control logic **110** converts the first access instruction to a second access instruction wherein the at least a first address within the logical address space is mapped to at least a first physical address corresponding to the mapping of the logical address space to the plurality of physical addresses. In S**540** the second access instruction is sent by the NIC over a second communication protocol to at least a storage server of the plurality of storage servers corresponding to the at least one of the plurality of physical addresses. The second communication protocol may implement, in some embodiments, remote direct memory access (RDMA) protocols, such as but not limited to RDMA over Converged Ethernet (RoCE), Infiniband, and iWARP. In another exemplary embodiment, the communication may be over a peer-to-peer (P2P) network.

The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPUs"), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

What is claimed is:

1. A system for enabling a client device seamless access to a plurality of remote storage devices connected to the client device via a communication network, comprising:

a network interface controller (NIC);

a processing circuitry; and

a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to:

receive a plurality of physical addresses by the network interface controller communicatively coupled to the client device and to a plurality of storage servers, each physical address, when received, not being associated with any logical address of the client device, each of the plurality of storage servers communicatively coupled to at least one storage device, the plurality of physical addresses enabling access by the NIC to the remote storage devices; and

generate for the client device by the network interface controller a single virtual storage device having a logical address space by mapping, by the network interface controller, each of the plurality of received physical addresses to at least a unique logical address of the virtual storage device; and

present, by the network interface controller, the single virtual storage device to the client to be used as a local storage device by the client.

2. The system of claim **1**, wherein the system is further configured:

receive a first access instruction having a first logical address within the logical address space, the first access instruction according to a first communication protocol;

convert by the controller the first address to a second access instruction according to a second communication protocol, wherein the first logical address is mapped to a physical address of the plurality of physical addresses; and

send the second access instruction over the second communication protocol to at least a storage server of the plurality of storage servers corresponding to the physical address.

3. The system of claim **2**, wherein the first communication protocol is NVMExpress.

4. The system of claim **2**, wherein the second communication protocol is an implementation of remote direct memory access (RDMA).

5. The system of claim **4**, wherein the implementation is any of: iWARP, Infiniband, or RDMA over Converged Ethernet (RoCE).

6. The system of claim **1**, wherein the system is further configured to:

send a request for an additional physical address; and

map a received additional physical address to a unique logical address of the logical address space.

7. The system of claim **1**, wherein the system is further configured to:

send a request for a plurality of additional physical addresses;

map a virtual address of the plurality of virtual addresses to a secondary virtual address of a secondary plurality of virtual addresses; and

map each additional physical layer to a unique secondary virtual address.

8. The system of claim **1**, wherein the single virtual storage is generated using input/output (I/O) virtualization.

9. The system of claim **1**, wherein the I/O virtualization is Single Root I/O Virtualization (SR-IOV).

10. The system of claim **9**, wherein the I/O virtualization generates a plurality of single virtual storages.

11. The system of claim **1**, wherein the system is further configured to:

receive a second plurality of physical addresses by the controller, the second plurality of physical addresses enabling access by the NIC to a local storage device communicatively coupled to the client device; and

map the second plurality of physical addresses to the logical address space.

* * * * *

EXHIBIT L

US010728331B2

(12) **United States Patent**
Romem et al.

(10) **Patent No.: US 10,728,331 B2**
(45) **Date of Patent: Jul. 28, 2020**

(54) **TECHNIQUES FOR DYNAMIC CACHE USE BY AN INPUT/OUTPUT DEVICE**

(71) Applicant: **Excelero Storage Ltd.**, Tel Aviv (IL)

(72) Inventors: **Yaniv Romem**, Jerusalem (IL); **Ofer Oshri**, Kfar Saba (IL); **Omri Mann**, Jerusalem (IL)

(73) Assignee: **EXCELERO STORAGE LTD.**, Tel Aviv (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 317 days.

(21) Appl. No.: **15/630,003**

(22) Filed: **Jun. 22, 2017**

(65) **Prior Publication Data**

US 2017/0374150 A1 Dec. 28, 2017

Related U.S. Application Data

(60) Provisional application No. 62/353,051, filed on Jun. 22, 2016.

(51) **Int. Cl.**

G06F 15/16	(2006.01)
H04L 29/08	(2006.01)
G06F 3/06	(2006.01)
H04L 29/06	(2006.01)
G06F 15/173	(2006.01)
G06F 12/0813	(2016.01)

(52) **U.S. Cl.**

CPC ***H04L 67/1097*** (2013.01); ***G06F 3/0604*** (2013.01); ***G06F 3/064*** (2013.01); ***G06F 3/067*** (2013.01); ***G06F 3/0631*** (2013.01); ***G06F 15/17331*** (2013.01); ***H04L 67/2842*** (2013.01); ***H04L 67/42*** (2013.01); *G06F 12/0813* (2013.01); *G06F 2212/154* (2013.01); *G06F 2212/163* (2013.01); *G06F 2212/502* (2013.01)

(58) **Field of Classification Search**

CPC G06F 11/34; G06F 12/0813; G06F 15/17331; G06F 2212/154; G06F 2212/163; G06F 2212/502; G06F 3/0604; G06F 3/0631; G06F 3/064; G06F 3/067; H04L 67/1097; H04L 67/2842; H04L 67/42

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,573,895 B2	8/2009	King		
7,685,254 B2 *	3/2010	Pandya	....	H04L 63/20 709/217
7,944,920 B2 *	5/2011	Pandya	....	H04L 29/06 370/392

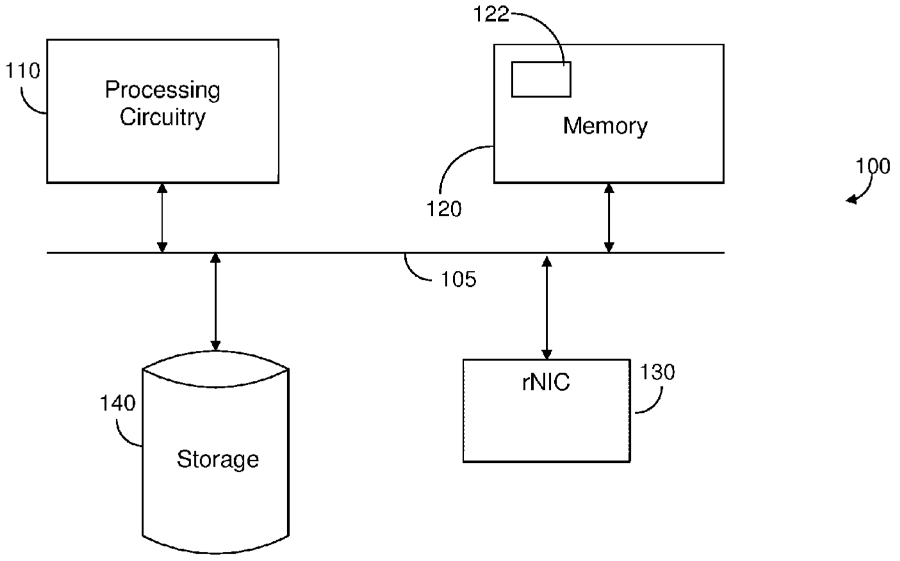
(Continued)

Primary Examiner — Moustafa M Meky
Assistant Examiner — Thorne E Waugh
(74) *Attorney, Agent, or Firm* — M&B IP Analysts, LLC

(57) **ABSTRACT**

A system and method for dynamic caching by a client device having remote memory access to a server. The system includes: a processing circuitry; and at least one memory, the at least one memory containing instructions that, when executed by the processing circuitry, configure the system to: configure a network interface of the client device to: request a memory allocation of at least a portion of a storage of the client device; receive, in real-time, the requested memory allocation of the client device storage; and store, in the allocated at least a portion of the client device storage, at least a first portion of cached metadata, wherein the cached metadata corresponds to at least an access operation between the client device and the server.

16 Claims, 4 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

9,311,110	B2 *	4/2016	Tamir	G06F 9/4416
9,467,512	B2 *	10/2016	Tamir	G06F 15/167
9,864,537	B2 *	1/2018	Chinnakkonda Vidyapoornachary	G11C 29/52
10,257,273	B2 *	4/2019	Govind	H04L 67/1097
2013/0198312	A1 *	8/2013	Tamir	G06F 15/167 709/212
2014/0143364	A1	5/2014	Guerin et al.	
2014/0214997	A1	7/2014	Metzler et al.	
2015/0012735	A1 *	1/2015	Tamir	G06F 9/4416 713/2
2015/0067088	A1	3/2015	Guerin et al.	
2017/0039145	A1 *	2/2017	Wu	G06F 12/0813

* cited by examiner

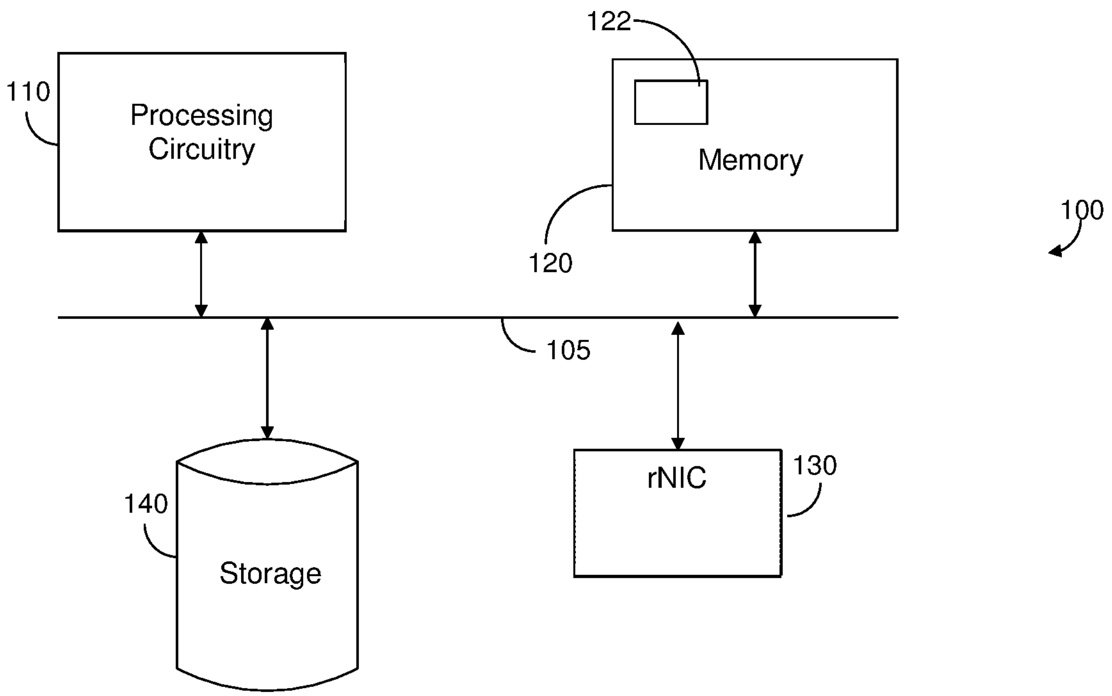

122
110
Processing Circuitry
Memory
100
120
105
130
rNIC
140
Storage

FIG. 1

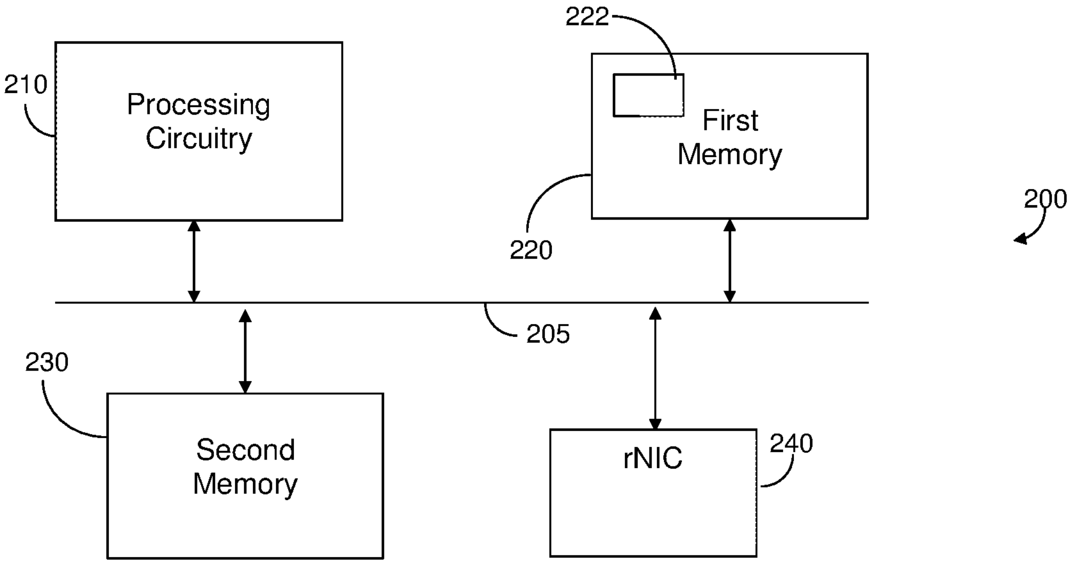

210
Processing Circuitry
222
First Memory
220
200
205
230
Second Memory
rNIC
240

FIG. 2

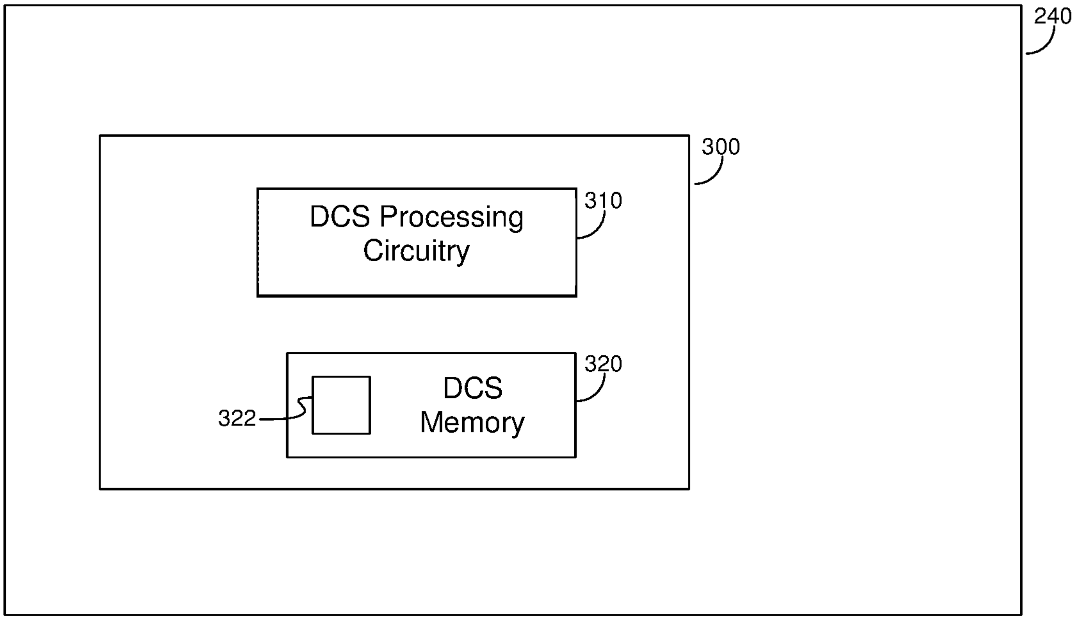


FIG. 3

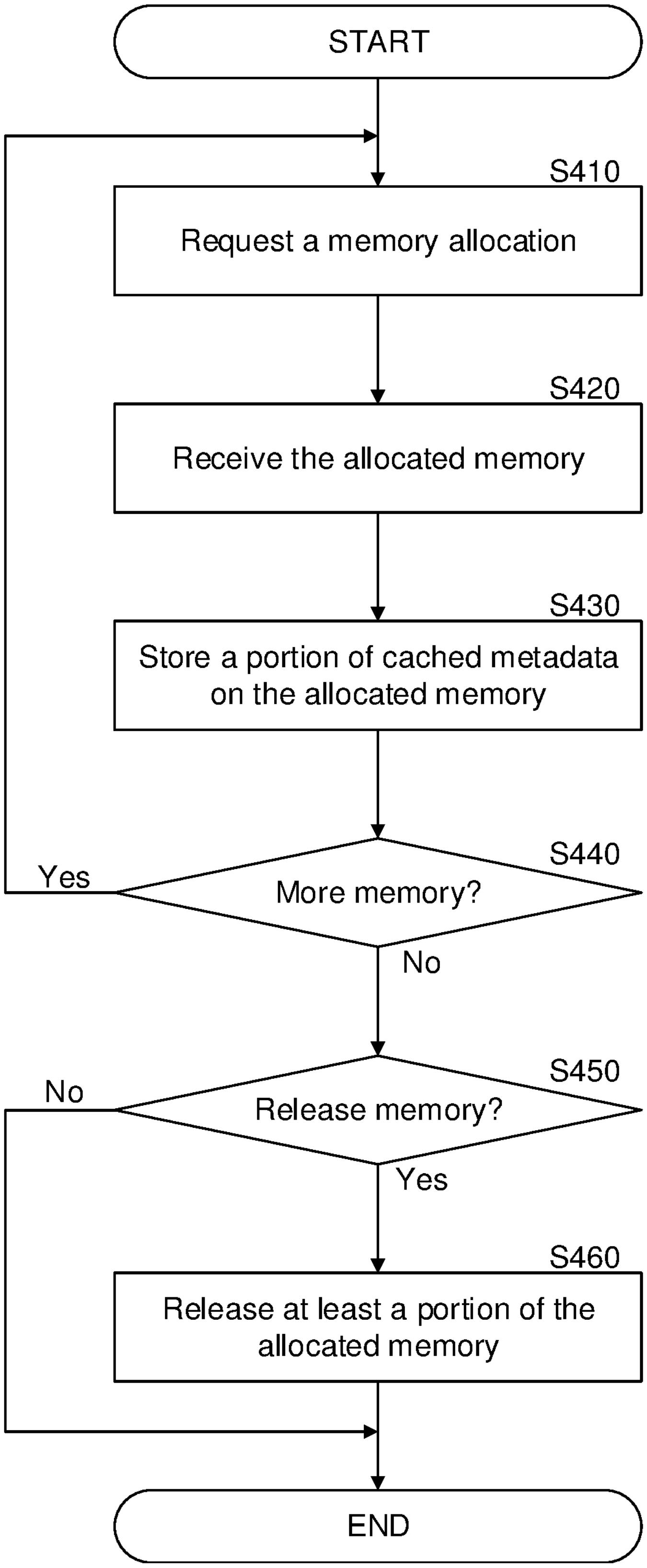

START
S410
Request a memory allocation
S420
Receive the allocated memory
S430
Store a portion of cached metadata on the allocated memory
S440
More memory?
Yes
No
S450
Release memory?
No
Yes
S460
Release at least a portion of the allocated memory
END

FIG. 4

TECHNIQUES FOR DYNAMIC CACHE USE BY AN INPUT/OUTPUT DEVICE

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 62/353,051 filed on Jun. 22, 2016, the contents of which are hereby incorporated by reference.

TECHNICAL FIELD

The present disclosure relates generally to accessing remote storage devices, and more particularly to efficiently accessing Remote Direct Memory Access devices.

BACKGROUND

Network accessible storages have and continue to become increasingly prevalent. Network accessible storages provide access to files and data without requiring local storage. A particular method for providing access to storages available over a network is remote direct memory access. Remote direct memory access allows for direct access of the memory of one computer by the memory of another directly and without involving either computer's operating system. This direct access permits high-throughput, low-latency networking as compared to less direct methods of remote access, especially when utilized in massively parallel computer clusters.

A challenge faced by network accessible storage devices is that such devices are typically slower to access than local storages. Further, aggregations of slowed accesses can result in severe underperformance when utilizing remote storage access than for local storage access, particularly when scaling up to incorporate a high number of computers accessible over a network. Thus, even when remote direct memory access is utilized, remote access may be significantly less efficient than local access.

It would therefore be advantageous to provide a solution that would overcome the challenges noted above by improving communications with network accessible storage devices.

SUMMARY

A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the term "some embodiments" or "certain embodiments" may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

Certain embodiments disclosed herein also include a system for dynamic caching by a client device having remote memory access to a server. The system includes: a processing circuitry; and at least one memory, the at least one memory containing instructions that, when executed by the processing circuitry, configure the system to: configure a network interface of the client device to: request a memory

allocation of at least a portion of a storage of the client device; receive, in real-time, the requested memory allocation of the client device storage; and store, in the allocated at least a portion of the client device storage, at least a first portion of cached metadata, wherein the cached metadata corresponds to at least an access operation between the client device and the server.

Certain embodiments disclosed herein also include a non-transitory computer readable medium having stored thereon causing a processing circuitry to execute a process, the process comprising: configuring a network interface of the client device to: request a memory allocation of at least a portion of a storage of the client device; receive, in real-time, the requested memory allocation of the client device storage; and store, in the allocated at least a portion of the client device storage, at least a first portion of cached metadata, wherein the cached metadata corresponds to at least an access operation between the client device and the server.

Certain embodiments disclosed herein also include a method for dynamic caching by a client device having remote memory access to a server. The method comprises: configuring a network interface of the client device to: request a memory allocation of at least a portion of a storage of the client device; receive, in real-time, the requested memory allocation of the client device storage; and store, in the allocated at least a portion of the client device storage, at least a first portion of cached metadata, wherein the cached metadata corresponds to at least an access operation between the client device and the server.

Certain embodiments disclosed herein also include a system for dynamic caching by a client device having remote memory access to a server. The system includes: a processing circuitry; and at least one memory, the at least one memory containing instructions that, when executed by the processing circuitry, configure the system to: configure a network interface of the client device to: request a memory allocation of at least a portion of a storage of the client device from a driver agent of the client device, wherein the driver agent is configured to: dynamically allocate at least a portion of a storage of the client device and to the network interface; and store, in the allocated at least a portion of the client device storage, at least a first portion of cached metadata, wherein the cached metadata corresponds to at least an access operation between the client device and the server.

BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

FIG. **1** is a schematic diagram of a server for providing remote direct memory access storage to a client device.

FIG. **2** is a schematic diagram of a client device configured for remote direct memory access.

FIG. **3** is a schematic diagram of a dynamic caching system according to an embodiment.

FIG. **4** is a method for dynamic caching by a client device having remote memory access to a server according to an embodiment.

DETAILED DESCRIPTION

It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

The various disclosed embodiments include a method and system for dynamic caching by a client device having remote memory access to a server. A memory allocation is requested from a storage of the client device. A dynamic allocation of at least a portion of memory of the client device storage is received. At least a first portion of cached metadata is stored in the dynamically allocated portion of memory. The cached metadata corresponds at least to an access operation between the client device and the server. A second portion of the cached metadata may be stored in a storage of an input/output device of the client device.

The embodiments disclosed herein may provide increased efficiency of remote memory access and, in particular, remote direct memory access, as compared to, for example, remote direct memory access without dynamic caching of metadata. Specifically, the disclosed embodiments may be utilized to improve access times and communications for remote direct memory access.

FIG. **1** shows an example schematic diagram illustrating a server **100** for providing remote direct memory access storage to a client device (e.g., the client device **200**, FIG. **2**). The server **100** includes a processing circuitry **110**, a memory **120**, a storage **130**, and an input/output (I/O) device such as a remote network interface controller (rNIC) **140**. In an embodiment, the components of the server **100** may be communicatively connected via a bus **105**.

The processing circuitry **110** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

The memory **120** may be volatile (e.g., RAM, etc.), non-volatile (e.g., ROM, flash memory, etc.), or a combination thereof. In one configuration, computer readable instructions to implement one or more embodiments disclosed herein may be stored in the storage **140**.

Alternatively or collectively, the memory **120** may be configured to store software.

Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the one or more processors, cause the processing circuitry **110** to provide remote direct memory access.

The memory **120** may further include a memory portion **122** containing the instructions for causing the processing circuitry **110** to provide remote direct memory access. The storage **140** may be magnetic storage, optical storage, and the like, and may be realized, for example, as flash memory or other memory technology, CD-ROM, Digital Versatile Disks (DVDs), or any other medium which can be used to

store the desired information. The storage **140** may store instructions for executing the methods as described herein.

The remote network interface controller **130** allows the server **100** to communicate with a network (not shown) for purposes such as, but not limited to, receiving data to be stored in the storage **140**, providing access to data stored in the storage **140** and the like. The communications via the network may therefore be utilized to provide remote direct memory access to data stored in the storage **140** by a client device (e.g., the client device **200**, FIG. **2**).

FIG. **2** shows an example schematic diagram illustrating a client device **200** configured for remote direct memory access storage. In an example implementation, the client device **200** is configured to communicate with the server **100**, FIG. **1**, over a network (not shown) to store and retrieve data via remote direct memory access. The client device **200** includes a processing circuitry **210**, a memory **220**, a second memory **230**, and an input/output (I/O) device such as a remote network interface controller (rNIC) **240**. In an embodiment, the components of the client device **200** may be communicatively connected via a bus **205**.

The processing circuitry **210** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

Each of the first memory **220** and the second memory **230** may be volatile (e.g., RAM, etc.), non-volatile (e.g., ROM, flash memory, etc.), or a combination thereof. The first memory **220** may store, e.g., instructions. The second memory **230** may be utilized by the client device **200** to store a buffer cache.

The first memory **220** may be configured to store software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the one or more processors, cause the processing circuitry **210** to perform remote direct memory access.

The first memory **220** may further include a memory portion **222** containing the instructions for causing the processing circuitry **210** to perform remote direct memory access. In some implementations, the first memory **220** may further include a second memory portion (not shown) allocated for use by an input/output device such as, but not limited to, the remote network interface controller **240**.

The remote network interface controller **240** allows the client device **200** to communicate with a network (not shown) for purposes such as, but not limited to, retrieving data stored via remote direct memory access, sending data to be stored via remote direct memory access, and the like. The remote network interface controller may be configured to perform remote direct memory access and to cache data in, e.g., the second memory **230**. To this end, in an embodiment, the remote network interface controller **240** may include or have disposed thereon a system for dynamic caching (e.g., the dynamic caching system **300**, FIG. **3**). In an example implementation, the caching system may be a system-on-chip included in the remote network interface controller **240**.

It should be noted that the embodiments described herein with respect to FIG. **2** are discussed as featuring a first memory **220** and a second memory **230** merely for simplicity purposes and without limiting the disclosed embodiments. The first memory **220** and the second memory **230** may be equally implemented as first and second portions of the same memory (not shown) without departing from the scope of the disclosure.

FIG. **3** is an example schematic diagram of a dynamic caching system **300** according to an embodiment. In the example schematic diagram shown in FIG. **3**, the dynamic caching system **300** is implemented as a system-on-a-chip (SoC) included in the remote network interface controller **240**, FIG. **2**.

In the example schematic diagram shown in FIG. **3**, the dynamic caching system **300** includes a dynamic caching system (DCS) processing circuitry **310** and a DCS memory **320**. The DCS processing circuitry **310** may be realized as one or more hardware logic components and circuits. The DCS memory **320** may be volatile (e.g., RAM, etc.), non-volatile (e.g., ROM, flash memory, etc.), or a combination thereof.

The DCS memory **322** may include a memory portion **322** containing cached metadata of remote direct memory access instructions sent between the client device **200** and the server **100**. In an embodiment, the cached metadata may be virtual block metadata including a map of a plurality of virtual addresses, each virtual address corresponding to either a unique secondary virtual address or to a unique physical address of a storage device.

In some implementations, the DCS memory **320** may not be sufficiently large to contain the require metadata. To this end, in some implementations, a driver agent (not shown) may be installed on a memory accessible to, e.g., an operating system of the client device **200** such as, for example, the second memory **230** of the client device **200**. The agent is configured to dynamically allocate the portions of the second memory **230** as required to store the cached metadata. The agent may be configured to release portions of the second memory **230** according to the needs of the client device **200** and the remote network interface controller **240**.

It should be noted that the dynamic caching system **300** is described herein as being implemented in the client device **200** merely for example purposes and without limiting the disclosed embodiments. In some implementations, the dynamic caching system may be implemented in, e.g., a remote server (not shown). In such implementations, the dynamic caching system **300** is not configured to cache metadata in the DCS memory **322**, and may cause caching of metadata by the client device **200** using, for example, the driver agent described herein above.

FIG. **4** is an example flowchart **400** illustrating a method for dynamic caching by a client device having remote memory access to a server according to an embodiment. In an embodiment, the method is for dynamically allocating memory of a cache to an input/output (I/O) interface such as a network interface controller (e.g., the remote network interface controller **240**, FIG. **2**) of a client device (e.g., the client device **200**, FIG. **2**) having access to a server (e.g., the server **100**, FIG. **1**). In an embodiment, the method is performed by the dynamic caching system **300**, FIG. **3**.

At S**410**, an allocation of at least a portion of memory of the client device is requested. In an embodiment, S**410** may include sending a request for allocation of system memory to an agent installed on the client device that is configured to dynamically allocate memory.

At S**420**, the requested allocation of memory is received. In an embodiment, S**420** includes receiving a block of addresses, each block corresponding to a portion of the allocated memory. The requested allocation may be performed by the agent installed on the client device.

At S**430**, at least a portion of metadata is stored in the allocated memory. In an embodiment, a first portion of the metadata may be stored in the allocated memory, and a second portion of the metadata may be stored in a secondary memory (e.g., a memory of the dynamic caching system **300**).

At S**440**, it is determined if additional memory is required and, if so, execution continues with S**410**; otherwise, execution continues with S**450**.

At S**450**, it is determined if at least a portion of the allocated memory should be released and, if so, execution continues with S**460**; otherwise, execution terminates. In an embodiment, if it is not determined that at least a portion of the allocated memory should be released, execution may continue with S**450**, thereby allowing for periodic or otherwise repeated checking whether the allocated memory should be released.

In an embodiment, S**450** may include determining whether a sufficient storage of the dynamic caching system is available for storing the cached metadata. In a further embodiment, when sufficient memory (e.g., of the DCS memory portion **322**) is available in the dynamic caching system for storing the cached metadata, the cached metadata may be stored therein and the corresponding allocated memory may be released for other use. In an embodiment, it is determined that there is sufficient memory when an amount of memory required to store the cached metadata is less than a combined total amount of memory of the dynamic caching system and the allocated memory.

In another embodiment, S**450** may include receiving a release instruction from a driver agent of the client device and, upon receiving the release instruction, releasing the allocated memory. In yet another embodiment, the allocated memory may be released when it is determined that requirements of the client device storage are prioritized over the cached metadata, thereby freeing the allocated memory for more important or otherwise higher priority tasks.

At S**460**, at least a portion of the allocated memory is released, thereby freeing the allocated memory for other storage. In an embodiment, S**460** may include synchronizing metadata between the allocated memory and a portion of a locally accessible storage. To this end, in a further embodiment, S**460** may include sending a request for metadata synchronization to the system hosting the remotely accessible storage in which the allocated memory is located. In yet a further embodiment, S**460** may include determining whether the cached metadata and the remotely accessible metadata match and, if not, synchronizing the metadata.

It should be noted that various embodiments are described herein as a system implemented (e.g., as a system-on-a-chip) in or on a network interface controller merely for example purposes and without limiting the disclosed embodiments. The embodiments disclosed herein are equally applicable to systems included in or on any other input/output (I/O) devices configured to perform direct memory access without departing from the scope of the disclosure. Such I/O devices may include a controller configured to request memory allocations from a processing circuitry of, e.g., a server configured for direct memory access.

The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPUs"), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

It should be understood that any reference to an element herein using a designation such as "first," "second," and so forth does not generally limit the quantity or order of those elements. Rather, these designations are generally used herein as a convenient method of distinguishing between two or more elements or instances of an element. Thus, a reference to first and second elements does not mean that only two elements may be employed there or that the first element must precede the second element in some manner. Also, unless stated otherwise, a set of elements comprises one or more elements.

As used herein, the phrase "at least one of" followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including "at least one of A, B, and C," the system can include A alone; B alone; C alone; A and B in combination; B and C in combination; A and C in combination; or A, B, and C in combination.

What is claimed is:

1. A system for dynamic caching for a client device having remote memory access to a server, comprising:

a processing circuitry; and

at least one memory, the at least one memory containing instructions that, when executed by the processing circuitry, configure the system to:

configure a network interface of the client device to:

request a memory allocation of at least a portion of a storage of the client device;

receive, in real-time, the requested memory allocation of at least a portion of the client device storage;

store, in the allocated at least a portion of the client device storage, at least a first portion of cached metadata,

wherein the cached metadata corresponds to at least an access operation between the client device and the server;

store a second portion of the cached metadata in at least a portion of the at least one memory; and

release at least a portion of the allocated client device storage, when it is determined that an amount of memory required to store the cached metadata is less than a combined total amount of memory of the system and the allocated client device storage.

2. The system of claim **1**, wherein the system is included in the network interface.

3. The system of claim **2**, wherein the network interface is a network interface controller, wherein the system is a system-on-a-chip deployed on the network interface controller.

4. The system of claim **1**, wherein the network interface is further configured to:

release at least a portion of the allocated client device storage, when a release instruction is received from a driver agent installed on the client device.

5. The system of claim **1**, wherein the network interface is further configured to:

release the allocated client device storage, when it is determined that requirements of the client device storage are prioritized over the cached metadata.

6. The system of claim **1**, wherein the network interface is further configured to:

synchronize the cached metadata stored in the client device and metadata stored in the server, when it is determined that the cached metadata and the metadata stored in the server do not match.

7. The system of claim **1**, wherein the requested memory allocation is received as a plurality of block addresses corresponding to a plurality of respective portions of the client device storage.

8. A non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to execute a process, the process comprising:

configuring a network interface of the client device to:

request a memory allocation of at least a portion of a storage of the client device;

receive, in real-time, the requested memory allocation of the at least a portion of the client device storage;

store, in the allocated at least a portion of the client device storage, at least a first portion of cached metadata, wherein the cached metadata corresponds to at least an access operation between the client device and the server;

store a second portion of the cached metadata in at least a portion of at least one memory of a dynamic caching system; and

release at least a portion of the allocated client device storage, when it is determined that an amount of memory required to store the cached metadata is less than a combined total amount of memory of the system and the allocated client device storage.

9. A method for dynamic caching by a client device having remote memory access to a server, comprising:

configuring a network interface of the client device to:

request a memory allocation of at least a portion of a storage of the client device;

receive, in real-time, the requested memory allocation of the at least a portion of the client device storage;

store, in the allocated at least a portion of the client device storage, at least a first portion of cached metadata, wherein the cached metadata corresponds to at least an access operation between the client device and the server;

store a second portion of the cached metadata in at least a portion of at least one memory of a dynamic caching system; and

release at least a portion of the allocated client device storage, when it is determined that an amount of memory required to store the cached metadata is less than a combined total amount of memory of the system and the allocated client device storage.

10. The method of claim **9**, wherein the network interface is a network interface controller, wherein the network interface controller is configured by a system-on-a-chip deployed on the network interface controller.

11. The method of claim **9**, further comprising:

configuring the network interface to:

release at least a portion of the allocated client device storage, when a release instruction is received from a driver agent installed on the client device.

12. The method of claim **9**, further comprising:

configuring the network interface to:

release the allocated client device storage, when it is determined that requirements of the client device storage are prioritized over the cached metadata.

13. The method of claim **9**, further comprising:

configuring the network interface to:

synchronize the cached metadata stored in the client device and metadata stored in the server, when it is determined that the cached metadata and the metadata stored in the server do not match.

14. The method of claim **9**, wherein the requested memory allocation is received as a plurality of block addresses corresponding to a plurality of respective portions of the client device storage.

15. A system for dynamic caching by a client device having remote memory access to a server, comprising:

a processing circuitry; and

at least one memory, the at least one memory containing instructions that, when executed by the processing circuitry, configure the system to:

configure a network interface of the client device to:

request a memory allocation of at least a portion of a storage of the client device from a driver agent of the client device, wherein the driver agent is configured to:

dynamically allocate at least a portion of a storage of the client device and to the network interface;

store, in the allocated at least a portion of the client device storage, at least a first portion of cached metadata, wherein the cached metadata corresponds to at least an access operation between the client device and the server;

store a second portion of the cached metadata in at least a portion of the at least one memory; and

release at least a portion of the allocated client device storage, when it is determined that an amount of memory required to store the cached metadata is less than a combined total amount of memory of the system and the allocated client device storage.

16. The system of claim **15**, wherein the network interface is further configured to:

release at least a portion of the allocated at least a portion of the client device storage, when a release instruction is received from the driver agent.

* * * * *

EXHIBIT M

US011374982B1

(12) **United States Patent**
Keren et al.

(10) **Patent No.: US 11,374,982 B1**
(45) **Date of Patent: Jun. 28, 2022**

(54) **STATIC ANALYSIS TECHNIQUES FOR DETERMINING REACHABILITY PROPERTIES OF NETWORK AND COMPUTING OBJECTS**

(71) Applicant: **Wiz, Inc.**, Palo Alto, CA (US)

(72) Inventors: **Shai Keren**, Ramat Gan (IL); **Daniel Hershko Shemesh**, Givat Shmuel (IL)

(73) Assignee: **Wiz, Inc.**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **17/179,135**

(22) Filed: **Feb. 18, 2021**

Related U.S. Application Data

(63) Continuation-in-part of application No. 17/109,883, filed on Dec. 2, 2020.

(51) **Int. Cl.**

G06F 15/173	(2006.01)
H04L 9/40	(2022.01)
G06F 16/901	(2019.01)
G06F 16/903	(2019.01)

(52) **U.S. Cl.**
CPC ***H04L 63/20*** (2013.01); ***G06F 16/903*** (2019.01); ***G06F 16/9024*** (2019.01); ***H04L 63/101*** (2013.01); ***H04L 63/1425*** (2013.01)

(58) **Field of Classification Search**
CPC ... H04L 63/20; H04L 63/101; H04L 63/1425; G06F 16/903; G06F 16/9024
USPC 709/201, 204, 205, 217, 218, 219, 223, 709/224
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,392,539	B2 *	6/2008	Brooks	H04L 63/0263 709/223
10,171,300	B2 *	1/2019	Eggen	H04L 41/0896
10,924,347	B1 *	2/2021	Narsian	H04L 41/0853
2004/0019803	A1	1/2004	Jahn	
2014/0157417	A1 *	6/2014	Grubel	H04L 63/1433 726/25
2016/0044057	A1	2/2016	Chenette et al.	
2016/0048556	A1 *	2/2016	Kelly	G06Q 30/02 707/767
2016/0359872	A1 *	12/2016	Yadav	H04L 63/1425
2016/0373944	A1 *	12/2016	Jain	H04W 24/02
2017/0075981	A1 *	3/2017	Carlsson	G06F 16/285
2018/0024981	A1 *	1/2018	Xia	G06F 40/18 715/215
2019/0095530	A1 *	3/2019	Booker	G06F 16/313
2020/0267175	A1	8/2020	Atighetchi et al.	

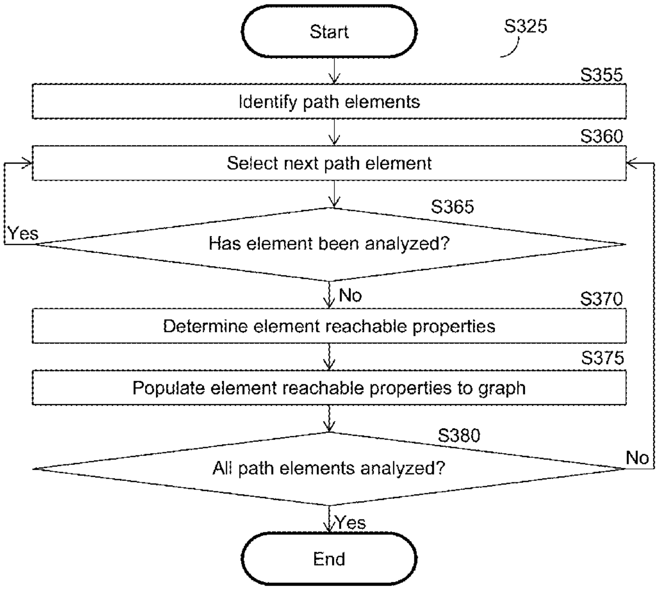
(Continued)

Primary Examiner — Liang Che A Wang
(74) *Attorney, Agent, or Firm* — M&B IP Analysts, LLC

(57) **ABSTRACT**

A method and system for determining reachability properties of security objects are provided. The method includes accessing a security graph, wherein the security graph lists all security objects and their connections in a cloud environment of an organization; identifying a plurality of network paths in the cloud environment, wherein each network path includes at least two security objects accessible in the cloud environment; for each of the plurality of identified network paths, iteratively analyzing each security object in a respective network path to determine its reachability properties, wherein the reachability properties of a security object as a minimal set of reachable properties of all other security objects in the respective network path; and populating the security graph with the determined reachability properties of each security object.

19 Claims, 9 Drawing Sheets


Start
S325
S355
Identify path elements
S360
Select next path element
S365
Yes
Has element been analyzed?
No
S370
Determine element reachable properties
S375
Populate element reachable properties to graph
S380
All path elements analyzed?
No
Yes
End

(56) References Cited

U.S. PATENT DOCUMENTS

2020/0322227	A1 *	10/2020	Janakiraman	H04L 43/045
2020/0374343	A1 *	11/2020	Novotny	G06F 17/12
2020/0382539	A1 *	12/2020	Janakiraman	H04L 63/0428

* cited by examiner

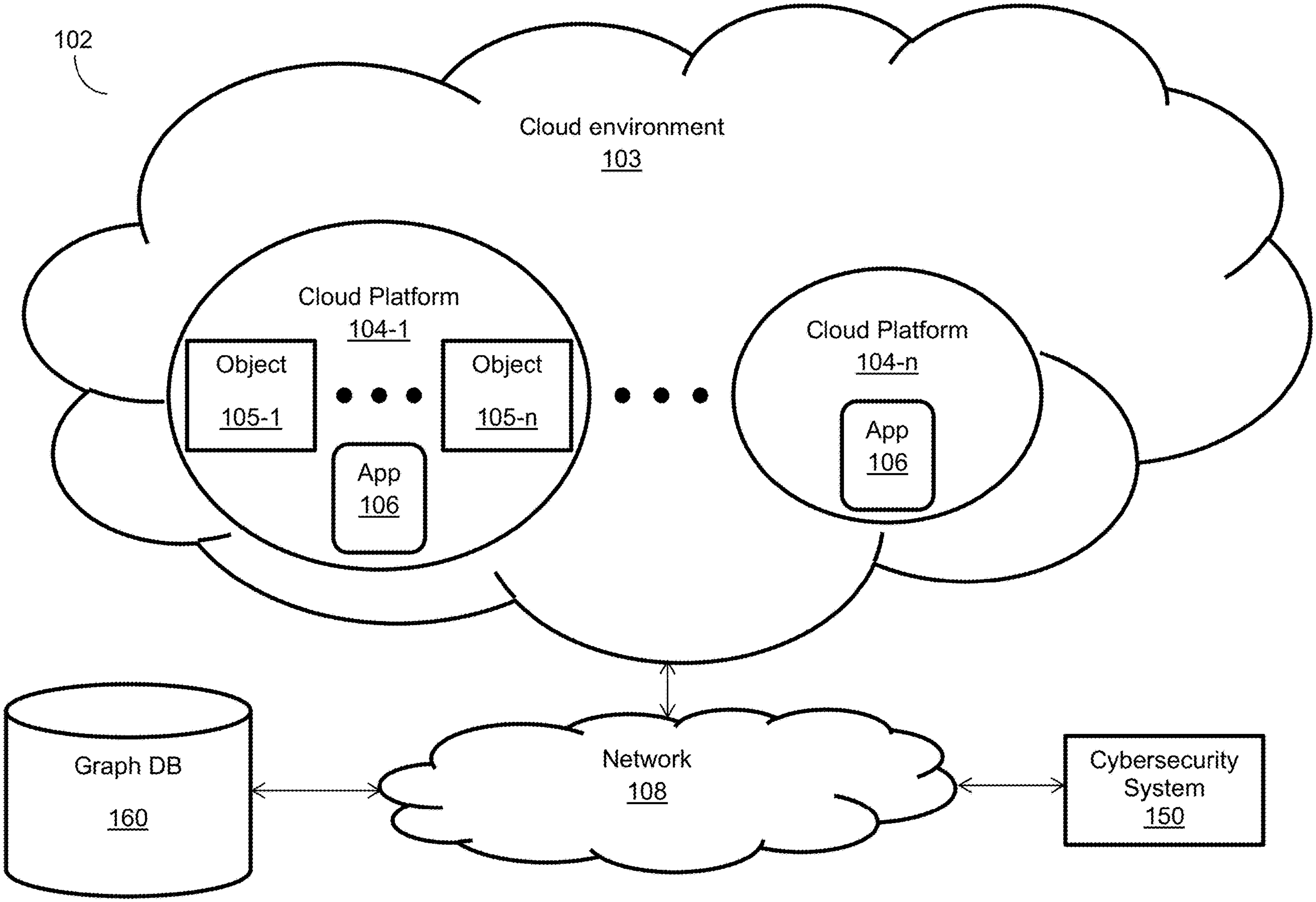

102
Cloud environment
103
Cloud Platform
104-1
Object
105-1
Object
105-n
App
106
Cloud Platform
104-n
App
106
Graph DB
160
Network
108
Cybersecurity
System
150

FIG. 1A

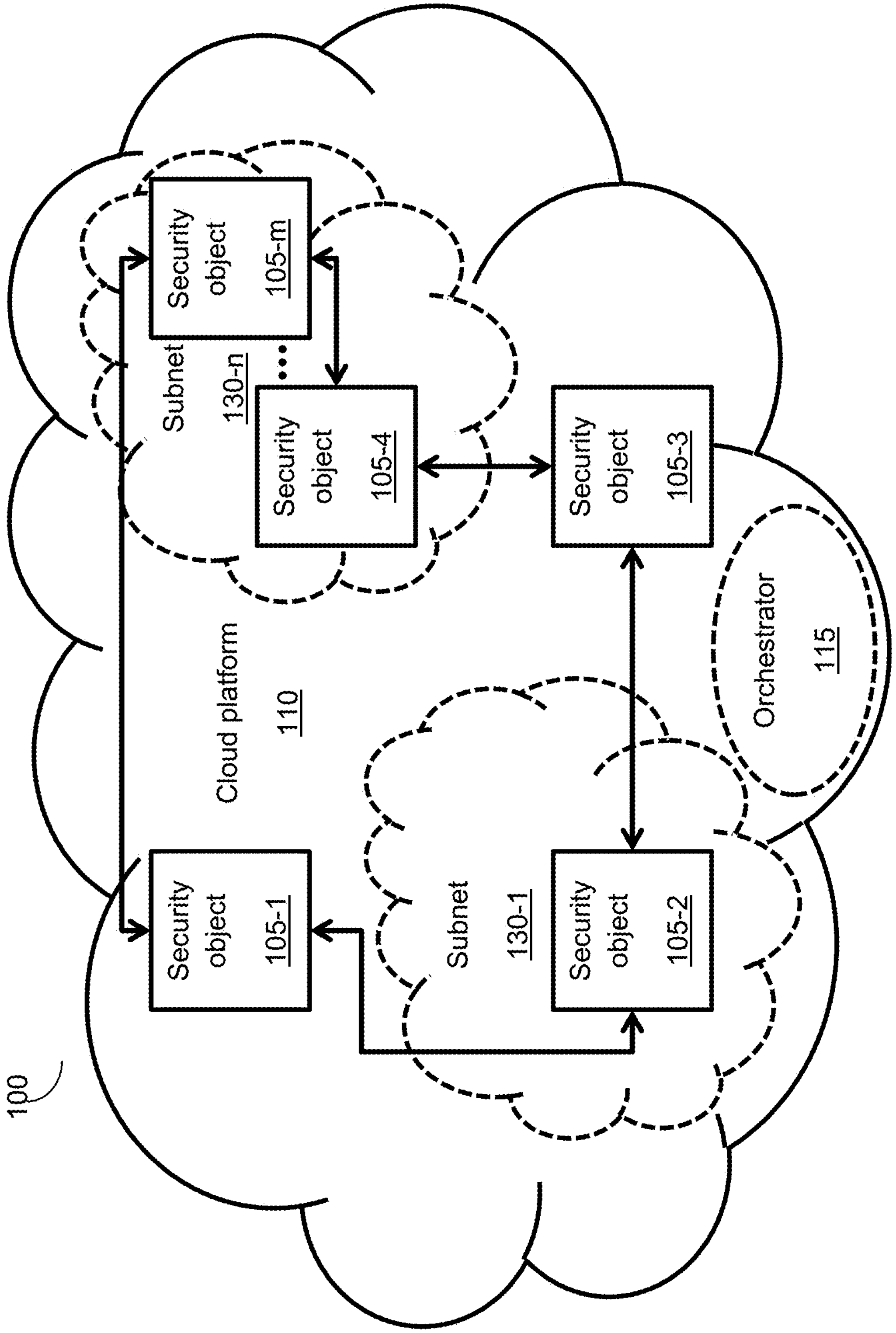

Security object 105-m
Subnet 130-n
Security object 105-4
Security object 105-3
Cloud platform 110
Orchestrator 115
Subnet 130-1
Security object 105-1
Security object 105-2
100

FIG. 1B

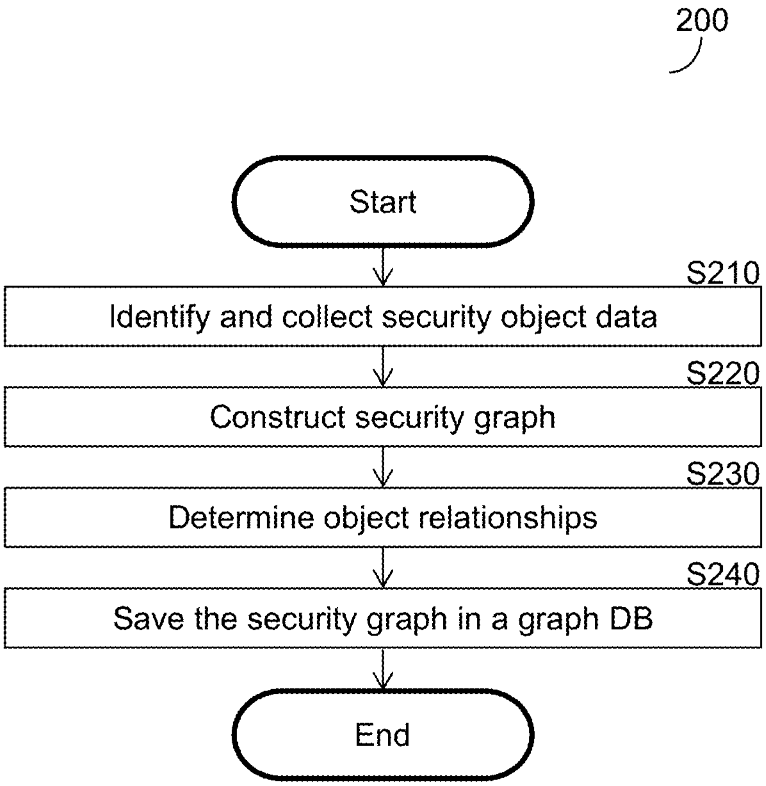

200
Start
S210
Identify and collect security object data
S220
Construct security graph
S230
Determine object relationships
S240
Save the security graph in a graph DB
End

FIG. 2

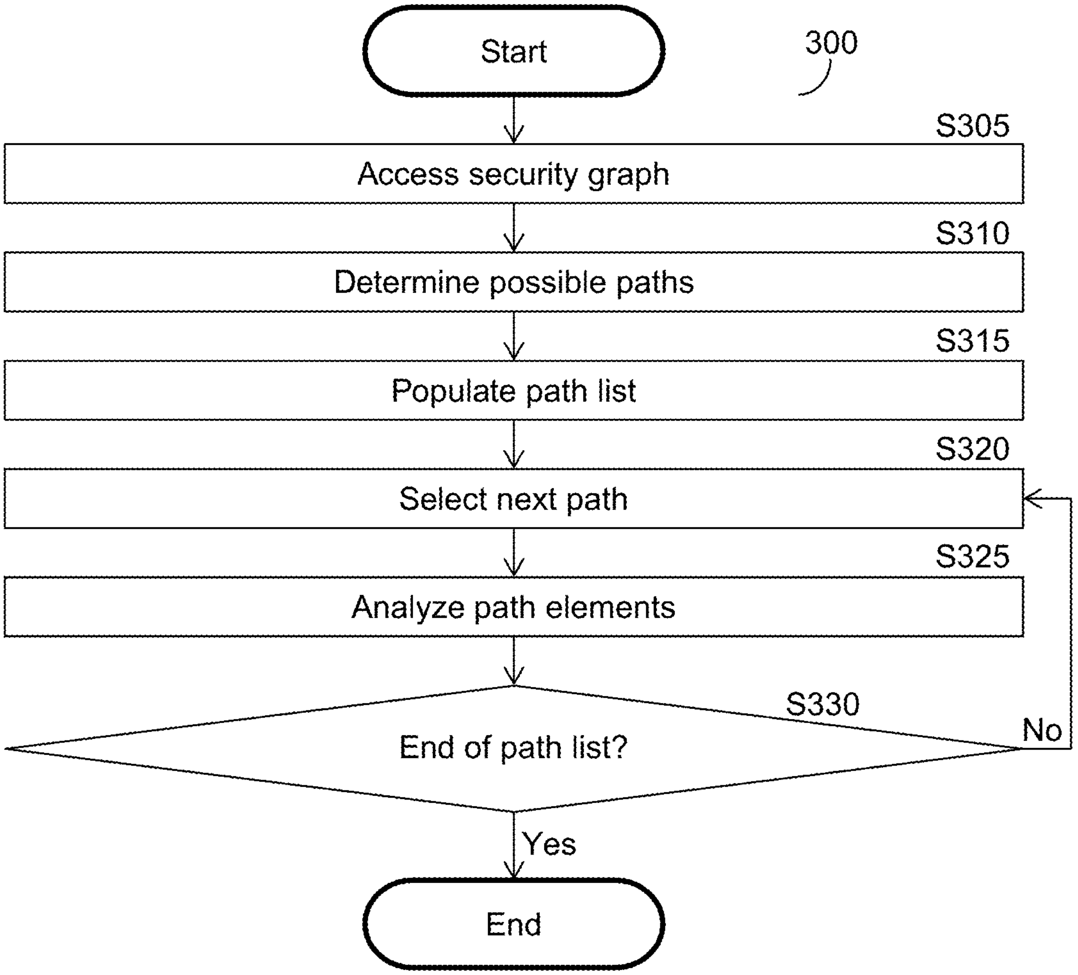

300
Start
S305
Access security graph
S310
Determine possible paths
S315
Populate path list
S320
Select next path
S325
Analyze path elements
S330
End of path list?
No
Yes
End

FIG. 3A

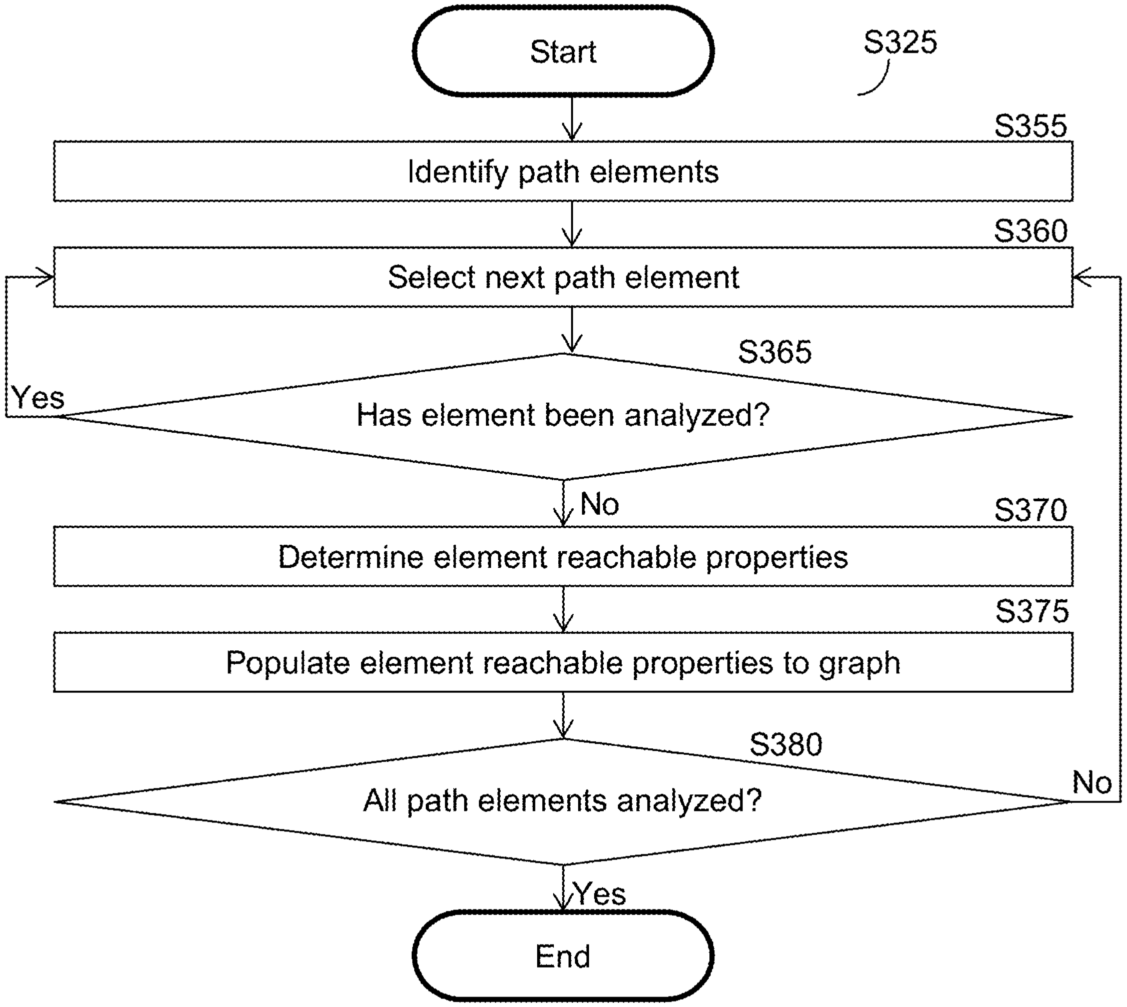

Start
S325
S355
Identify path elements
S360
Select next path element
S365
Has element been analyzed?
Yes
No
S370
Determine element reachable properties
S375
Populate element reachable properties to graph
S380
All path elements analyzed?
No
Yes
End

FIG. 3B

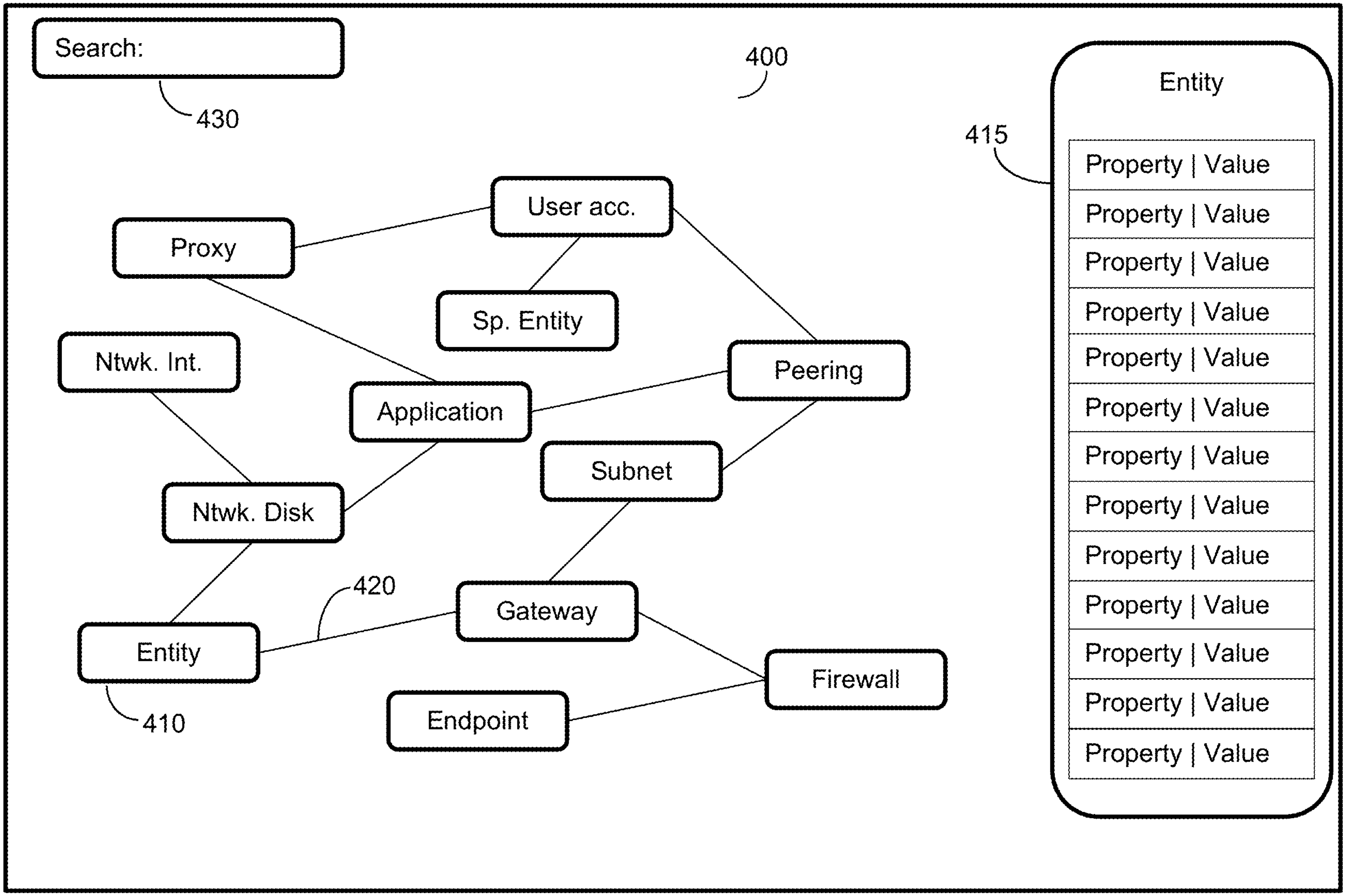
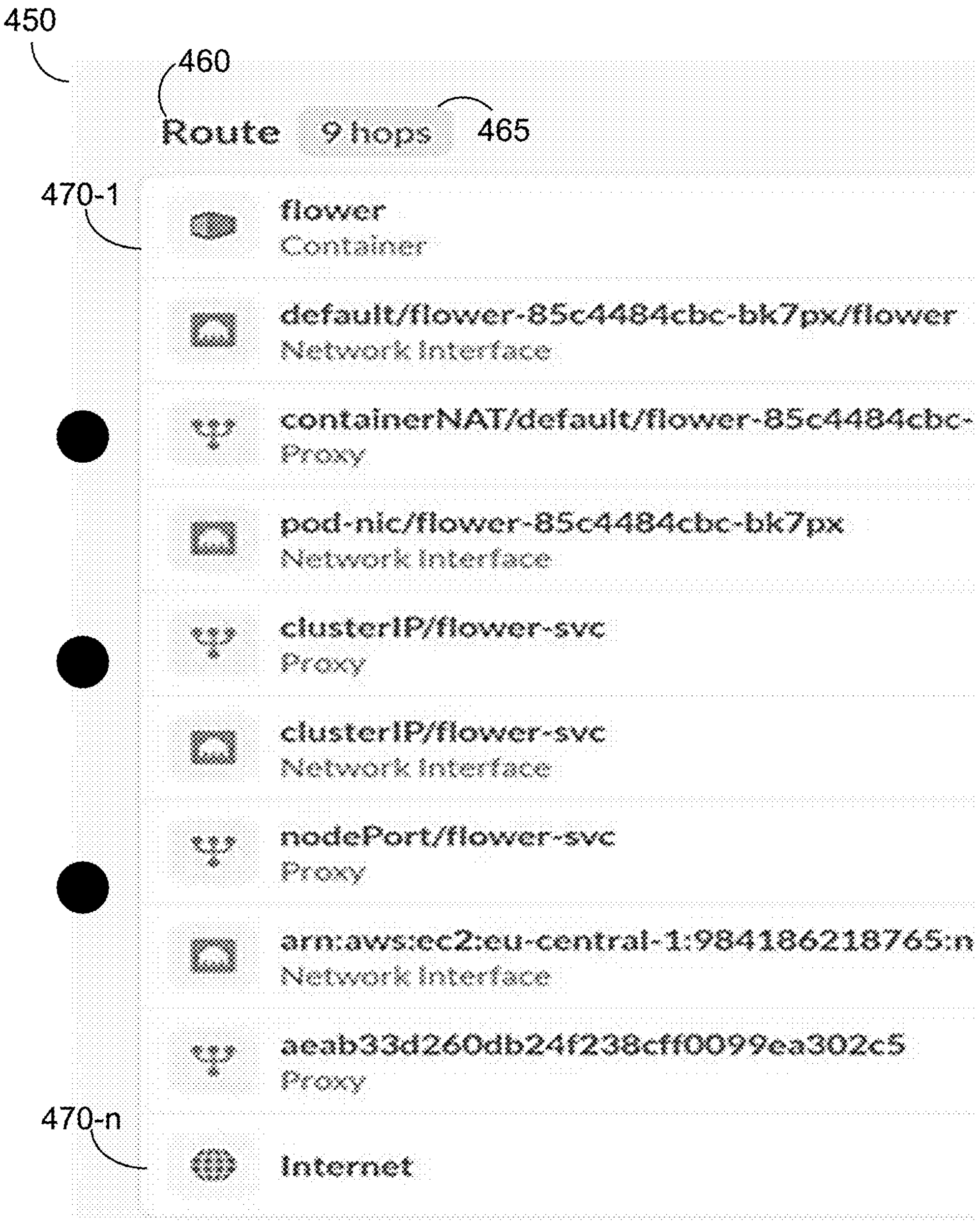

Search:
430
400
415
Entity
Property | Value
Property | Value
Property | Value
Property | Value
Property | Value
Property | Value
Property | Value
Property | Value
Property | Value
Property | Value
Property | Value
Property | Value
Property | Value
User acc.
Proxy
Sp. Entity
Ntwk. Int.
Application
Peering
Subnet
Ntwk. Disk
420
Gateway
Entity
Firewall
Endpoint
410

FIG. 4A



Route 9 hops

flower
Container

default/flower-85c4484cbc-bk7px/flower
Network Interface

containerNAT/default/flower-85c4484cbc-
Proxy

pod-nic/flower-85c4484cbc-bk7px
Network Interface

clusterIP/flower-svc
Proxy

clusterIP/flower-svc
Network Interface

nodePort/flower-svc
Proxy

arn:aws:ec2:eu-central-1:984186218765:n
Network Interface

aeab33d260db24f238cff0099ea302c5
Proxy

Internet

FIG. 4B

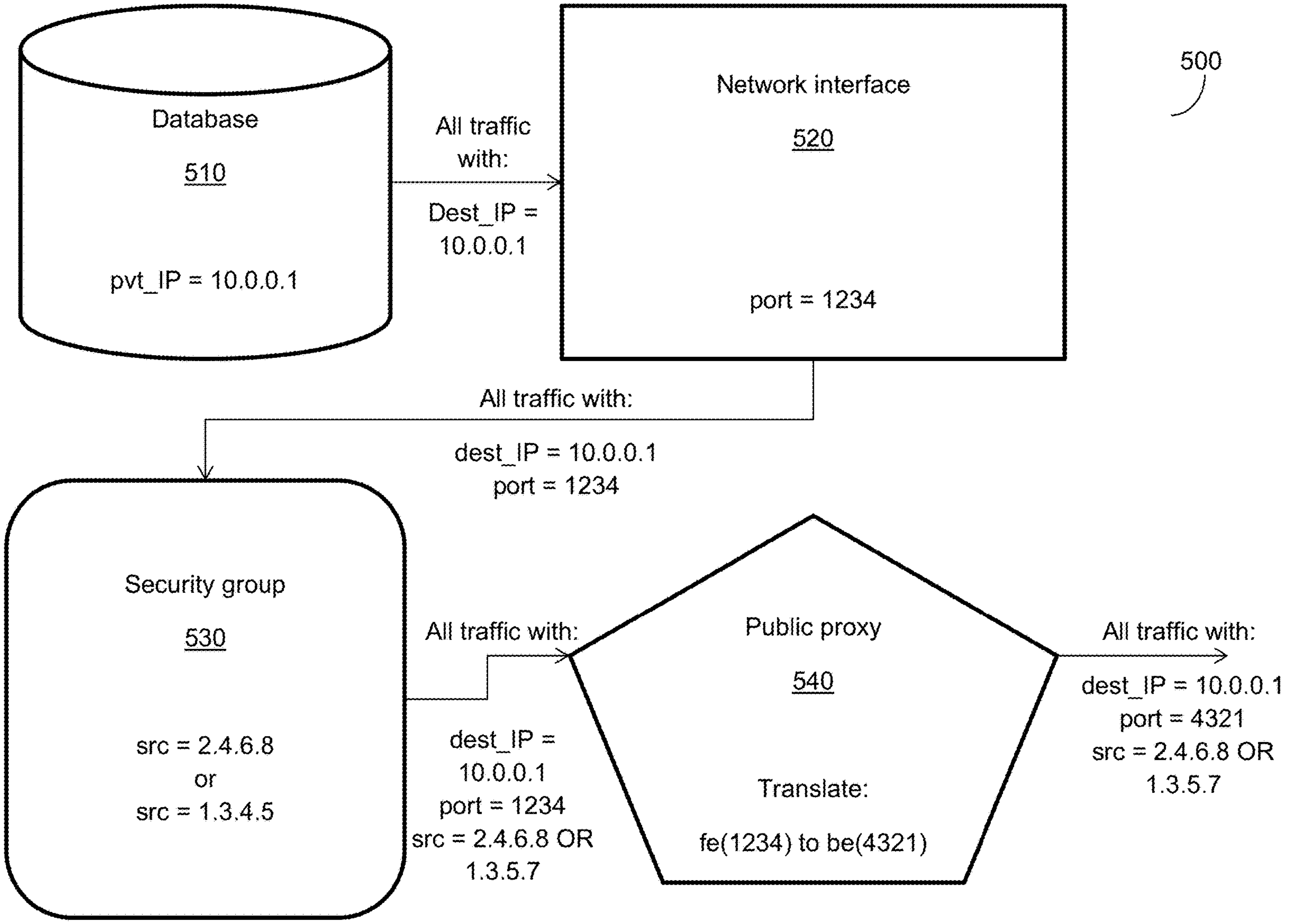

500
Database
510
pvt_IP = 10.0.0.1
All traffic with:
Dest_IP = 10.0.0.1
Network interface
520
port = 1234
All traffic with:
dest_IP = 10.0.0.1
port = 1234
Security group
530
src = 2.4.6.8
or
src = 1.3.4.5
All traffic with:
dest_IP = 10.0.0.1
port = 1234
src = 2.4.6.8 OR 1.3.5.7
Public proxy
540
Translate:
fe(1234) to be(4321)
All traffic with:
dest_IP = 10.0.0.1
port = 4321
src = 2.4.6.8 OR 1.3.5.7

FIG. 5

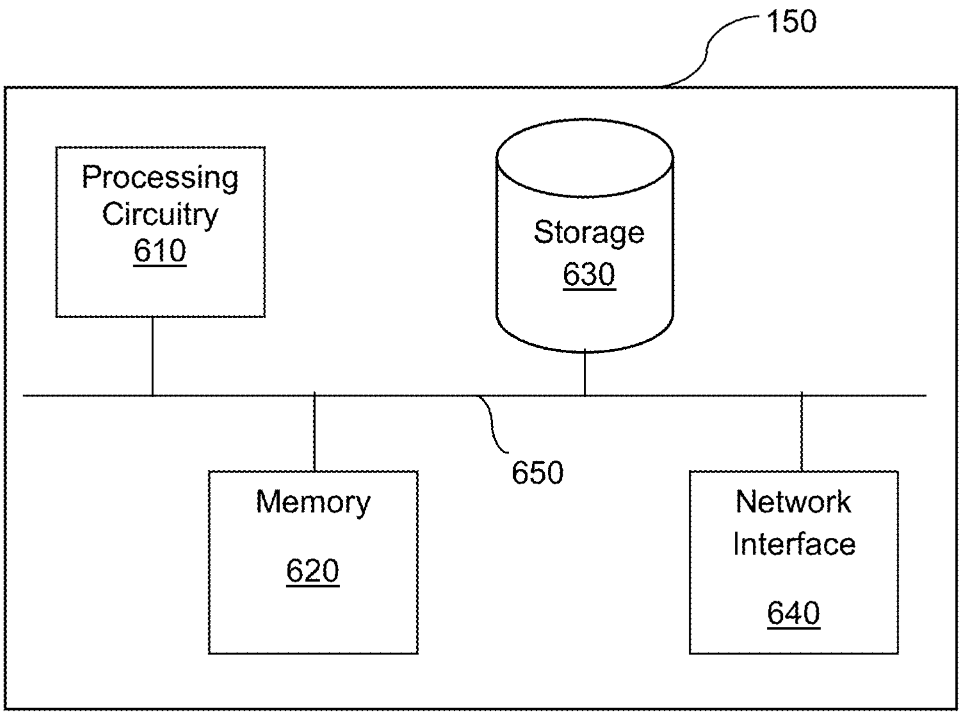

150
Processing Circuitry 610
Storage 630
650
Memory 620
Network Interface 640

FIG. 6

STATIC ANALYSIS TECHNIQUES FOR DETERMINING REACHABILITY PROPERTIES OF NETWORK AND COMPUTING OBJECTS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation-in-part application of U.S. patent application Ser. No. 17/109,883, filed Dec. 2, 2020, the contents of which are hereby incorporated by reference.

TECHNICAL FIELD

The present disclosure relates generally to network administration, in particular, to systems and methods for automated generation of network analysis insights.

BACKGROUND

As businesses, governments, and other organizations expand and increase their digital presence through various computer, network, and web technologies, the same parties may be increasingly vulnerable to developing cyber-threats. While updated solutions provide for management of prior cyber-threats, the same systems may include new vulnerabilities, which attackers may seek to identify and exploit to gain access to sensitive systems and data, Specifically, as organizations transition into multi-level computing systems, implementing computing solutions at the individual, group, team, and cloud levels, these systems, and the links between the elements of the layers, as well as the links between elements of different layers, include vulnerabilities which prior solutions fail to address.

Due to the distributed nature of large, multi-layered network systems, management of network access and use may be difficult or impossible for lone administrators or teams of administrators. Management of such code-to-cloud systems, and protection of the same, may require monitoring of large numbers of devices, systems, and components. Further, as each device, system, or component of a network system may be variously connected with the other elements of the system, including connections with multiple other devices via multiple protocols, management and monitoring of individual devices and connections may be untenable.

To address the need to manage large, distributed network systems, operators and administrators may employ various solutions to provide for network analysis. Certain network analysis solutions include manual review of devices, connections, and networks, providing for thorough, specific analysis of individual elements of a network. However, such manual solutions may require prohibitive outlays of time and effort to successfully review every component and connection of a large, multi-layer network, thus failing to provide a solution for analysis of modern network systems. In addition, various analysis solutions include solutions directed to the monitoring of specific device types, such as, for example, firewall control systems, which may provide for management of all firewalls installed in a given network. Similarly, protocol-specific analysis solutions may provide for monitoring of all traffic occurring over given protocols, within the network. However, such specialized solutions may fail to provide for streamlined monitoring and management of all components and connections of a network, where the network includes multiple types of devices communicating via multiple protocols. Further, protocol-agnostic solutions may provide for overall traffic management, providing monitoring and management solutions for all traffic arising within a network. However, such protocol-agnostic solutions may be over-broad, providing irrelevant or redundant information, and may require specification of connections to monitor, reducing efficacy in network-management contexts, while failing to provide device-specific insights, thereby failing to provide for integrated device and connection analysis within a complex, multi-layer network.

In addition, certain solutions providing for the management of large, distributed network systems may fail to provide for agentless management, non-logging solutions, and the like. Agentless management, whereby such large, distributed network systems are managed without the use of a dedicated management agent system or device, may provide for reduced maintenance requirements, as a management agent may require operation and maintenance in addition to the efforts required by the remainder of the network. In addition to failing to provide for agentless management, various solutions for the management of large, distributed network systems fail to provide for non-logging management of the same. Non-logging management, where network analyses and other management processes are executed without reference to netflow logs, provides for reductions in management computing requirements and resource dependency when compared with logging solutions, which may require, without limitation, the execution of additional processing steps or tasks to analyze or process netflow logs, the dependency of the management solution or process on various netflow log resources or repositories, and the like. In addition to the shortcomings described above, current solutions for management of large, distributed network systems may fail to provide for agentless, non-logging management.

Further, certain current solutions, as described hereinabove, fail to provide for the need to collect security object properties, and insights relevant thereto. As the systems described hereinabove, such as large networks implemented for a business or other organization, may include large numbers of devices, devices with complex configuration or management constraints current network analysis solutions may fail to provide granular, comprehensible representations of network information. Although current solutions may provide for the inspection of various security objects, thereby providing an understanding of the properties of individual inspected elements, such solutions fail to provide network information which describes the role of an individual entity within such a network or a cloud environment, as well as the vulnerabilities presented by the entities' placements within the network configuration. As a result, network administrators, and the like, may seek a solution which allows for network analysis and includes granular, comprehensive representation of security object properties and insights.

It would therefore be advantageous to provide a solution that would overcome the challenges noted above.

SUMMARY

A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the terms "some embodiments" or "certain embodiments" may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

Certain embodiments disclosed herein include a method for determining reachability properties of security objects. The method comprises accessing a security graph, wherein the security graph lists all security objects and their connections in a cloud environment of an organization; identifying a plurality of network paths in the cloud environment, wherein each network path includes at least two security objects accessible in the cloud environment; for each of the plurality of identified network paths, iteratively analyzing each security object in a respective network path to determine its reachability properties, wherein the reachability properties of a security object as a minimal set of reachable properties of all other security objects in the respective network path; and populating the security graph with the determined reachability properties of each security object.

In addition, certain embodiments disclosed herein include a system for determining reachability properties of security objects. The system comprises: a processing circuitry; and a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: access a security graph, wherein the security graph lists all security objects and their connections in a cloud environment of an organization; identify a plurality of network paths in the cloud environment, wherein each network path includes at least two security objects accessible in the cloud environment; for each of the plurality of identified network paths, iteratively analyze each security object in a respective network path to determine its reachability properties, wherein the reachability properties of a security object as a minimal set of reachable properties of all other security objects in the respective network path; and populate the security graph with the determined reachability properties of each security object.

BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

FIG. **1**A is a diagram of a cloud environment utilized to describe the various embodiments.

FIG. **1**B is a network diagram depicting a network system and various associated network and external objects, according to an embodiment.

FIG. **2** is a flowchart depicting a method for constructing security graphs, including network and computing objects, for cloud environments, according to an embodiment.

FIG. **3**A is a flowchart depicting a method for determining reachable properties of network and computing objects according to an embodiment.

FIG. **3**B is a flowchart depicting a process for analyzing a path to determine reachable properties according to an embodiment.

FIG. **4**A is a security graph visualization, generated according to an embodiment.

FIG. **4**B is a security graph object list, configured to provide information describing object-to-object routing within a security graph, according to an embodiment.

FIG. **5** is an illustration path, depicting the determination of reachable properties according to an embodiment.

FIG. **6** is a hardware block diagram depicting a cybersecurity system, according to an embodiment.

DETAILED DESCRIPTION

It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

The systems and methods described herein may be applicable to various systems, devices, networks, environments, layers, and the like, as well as cross-connections or multi-entity connections as may be established therebetween. The disclosed systems and methods may be applicable to provide support for various network features including, without limitation, application-layer communications, cloud-native constructs, cross-cloud and Kubernetes-to-cloud communications, third-party features, such as third-party containers and objects, container-management systems, such as Kubernetes, as may be virtualized as cloud objects, and the like, as well as any combination thereof.

FIG. **1**A is an example diagram **100** of a cloud environment **103** utilized to describe the various embodiments. A cloud environment **103** represents an organization's cloud-based resources, and the various connections between such resources. The cloud environment **103** may include a number of cloud computing platforms, **104-1** through **104-*n*** (hereinafter, "cloud platforms" **104** or "cloud platform" **104**), where a cloud platform may include multiple security objects, **105-1** through **105-*n*** (hereinafter, "security objects" **105** or "security object" **105**), one or more applications (collectively referred to as applications or apps **106**), and the like, as well as any combination thereof. Further, the cloud environment may be configured to connect, via a network **108**, with a cyber-security system **150** and a graph database (graph DB) **160** for one or more purposes including, without limitation, those described hereinbelow. As is applicable to the cloud platforms **104** and security objects **105**, "n" is an integer having a value greater than or equal to two. Further, it may be understood that, while a single configuration of a cloud environment **103** is shown for purposes of simplicity, a cloud environment **103** may include various combinations of platforms **104**, objects **105**, applications **106**, and the like, as well as any combination thereof, without loss of generality or departure from the scope of the disclosure.

A cloud platform **104** is a platform, architecture, or other, like, configuration providing for connectivity of the various objects **106**, applications **106**, and other, like, elements included in a cloud platform **104**, as well as the execution of various processes, instructions, and the like. A cloud platform **104** may be a commercially-available cloud system, provided on a service basis, such as, as examples and without limitation, Amazon AWS®, Microsoft Azure®, and the like. A cloud platform **104** may be a private cloud, a public cloud, a hybrid cloud, and the like. In addition, a cloud platform **104** may include, without limitation, container orchestration or management systems or platforms such as, as an example and without limitation, a Kubernetes® deployment, and the like, as well as any combination thereof.

A cloud platform **104** may be implemented as a physical network of discrete, interconnected objects, and the like, a virtual network, providing for interconnection of various virtual systems and devices, as well as a hybrid physical-virtual network, including both physical and virtualized components. A cloud platform **104** may be, or may replicate or otherwise simulate or emulate, as examples, and without limitation, a local area network, a wide area network, the Internet, the World-Wide Web (WWW), and the like, as well as any combination thereof. Further, a cloud platform **104** may include one or more subnets, such as the subnets, **130**, of FIG. **1**B, below, wherein each subnet may be configured to serve as a cloud platform **104** for the various security objects which may be included in the subnet, while retaining the connectivity and functionalities provided by the cloud platform **104**.

Security objects **105**, as may be included in a cloud platform **104**, are objects, systems, devices, components, applications, entities, and the like, configured to operate within the cloud platform **104** and provide various functionalities therein. Specifically, the security objects **105** may be objects configured to send, receive, or both send and receive, network data. The security objects **105** may be configured to connect with various other security objects **105**, various external objects, and the like, as well as any combination thereof, for purposes including, without limitation, sending data, receiving data, monitoring data transmissions, monitoring network status and activity, and the like, as well as any combination thereof. Security objects **105** may include one or more types of object, including, without limitation, network objects, computing objects, and the like, as well as any combination thereof. Network objects, as may be represented as security objects **105**, are objects configured to provide one or more traffic-management functionalities. Examples of network objects include, without limitation, proxies, firewalls, routing tables, and the like. Further, computing objects, as may be represented as security objects **105**, are objects configured to provide one or more data management or processing functionalities. Examples of computing objects include, without limitation, dedicated processing systems, fileservers, databases, and the like.

Examples of security objects **105**, as may be relevant to the methods, processes, and descriptions provided herein include, without limitation, objects providing support for application-layer communications and systems, including application-layer communications and systems relevant to layer seven of the open systems interconnection (OSI) model. Further examples of security objects **105**, relevant to the methods, processes, and descriptions provided herein, include, without limitation, cloud-native constructs, such as private endpoints, transit gateways, tag-based rulesets and objects configured to apply such rules, Kubernetes Istio and Calico services and applications, and the like. In addition, examples of security objects **105** may include, without limitation, third-party containers and images, such as Nginx, web-access firewall (WAF), and firewall implementations, multi-object or cross-object connections, such as cross-cloud connections and Kubernetes-to-cloud connections, as well as container managers, such as Kubernetes, and connections therewith. It may also be understood that security objects **105** may include other objects similar to those described hereinabove, as well as any combination thereof. As another example, security objects may include virtual entities, devices, and the like, to process layer-7 (application layer) traffic, such as objects relevant to Amazon AWS® layer seven services and applications, Amazon Load Balancer® (ALB) layer seven services and applications, Kubernetes ingress, and the like.

The security objects **105** may be configured to include one or more communication ports, where the included communication ports provide for connection of various objects according to one or more protocols, and at different communication layers of the OSI model.

In an example configuration, the security objects **105** are virtual entities or instances of systems, devices, or components, including virtual systems, devices, or components, or any combination thereof. Examples of security objects **105** include, without limitation, virtual networks, firewalls, network interface cards, proxies, gateways, containers, container management objects, virtual machines, subnets **130**, hubs, virtual private networks (VPNs), and the like, as well as any combination thereof.

The applications **106**, as may be executed in one or more cloud platforms **104**, are services, processes, and the like, configured to provide one or more functionalities by execution of various commands and instructions. The applications **106** may be part of a software project of an enterprise or organization. The applications **160** may interact or communicate with other applications, regardless of the platform **104** in which the applications **106** are deployed. It should be understood that a single application, including the same application, may be both present and executed in multiple cloud platforms **104**, including multiple cloud platforms **104** of the same cloud environment **103**, without loss of generality or departure from the scope of the disclosure.

The network **108** is a communication system providing for the connection of the cloud environment **103**, and its various components and sub-parts, with a cyber-security system **150**, as well as other, like, systems, devices, and components, and any combination thereof. The network **108** may be implemented as a physical network of discrete systems, devices, components, objects, and the like, a virtual network, providing for interconnection of various virtual systems and devices, as well as a hybrid physical-virtual network, including both physical and virtualized components. The network **108** may be, as examples, and without limitation, a local area network, a wide area network, the Internet, the World-Wide Web (WWW), and the like, as well as any combination thereof.

The cyber-security system **150** is a system, device, or component, configured to provide one or more network analysis functionalities including, without limitation, network analysis, traffic analysis, object querying, graph generation, and the like, as well as any combination thereof. The cyber-security system **150** may be configured to execute one or more instructions, methods, processes, and the like, including, without limitation, the processes described with respect to FIGS. **2** and **3**, other, like, processes, and any combination thereof.

The cyber-security system **150** may be configured as a physical system, device, or component, as a virtual system, device, or component, or in a hybrid physical-virtual configuration. A detailed description of a cyber-security system, **150**, according to an embodiment, is provided with respect to FIG. **6**, below. It may be understood that, while the cyber-security system **150** is depicted in FIG. **1**A as a discrete element external to the cloud environment **103**, the cyber-security system **150** may be included within any of the various elements of the network system **102**, including the cloud environment **103**, the various cloud platforms **104**, and subparts thereof, and the network **108**, without loss of generality or departure from the scope of the disclosure.

The graph database (graph DB) **160** is a storage or memory component, device, system, or the like, configured to provide one or more functionalities relevant to storage of graph-related data. The graph DB **160** may be configured to store graph-related data features of one or more types or formats, including, without limitation, raw data, graphs, graph edges, graph vertices, graph schemas, and the like, as well as any combination thereof, including those types or formats described hereinbelow.

Graphs, as may be included in the graph DB **160**, are data features including one or more graph vertices, where such graph vertices may be variously interconnected by one or more graph edges. Graphs may be configured to provide for one or more representations of various data sets, including, without limitation, presentation of network data according to one or more graph schemas, as described hereinbelow. As an example, a graph relevant to the description of a collection of interconnected security objects may include one or more graph vertices, where each graph vertex corresponds with a security object, and graph edges between such vertices, the edges corresponding with connections between the various security objects. Graphs, and elements thereof, may be generated based on one or more data sets, including during the execution of a graph generation process, such as is described with respect to FIG. **2**, below.

The graph DB **160** may be configured to store one or more graphs, graph, related data features, and the like, as well as any combination thereof. Graphs, as may be stored in the graph DB **160**, may be configured to include one or more functional attributes, including, without limitation, directionality, labeling, and the like, where such functional attributes may provide for the execution or enhancement of one or more processes or methods which would be inapplicable to a graph not including such functional attributes. A graph including directionality may be configured to include connection between graph nodes or vertices, via graph edges, as described herein, where the edges connecting such vertices may be uni-directional or bi-directional, providing for enhanced analysis of security object structures and relationships. Further, a graph configured to include labeling functionality may be configured to provide for the labeling of graph vertices, graph edges, or both, with one or more labels describing the various properties of the labeled vertices or edges. As an example, a graph vertex representing a virtual machine (VM) may be configured to include a "name" label, describing a name property of the VM. Further, the VM may be configured to run a container entity, where the container entity, as represented in the graph, may be respectively labeled. In addition, according to the same example, the connection, or edge, between the vertices representing the VM and the container entity may be uni-directional and may be labeled as a "run" edge, providing for analysis of the relationship between the vertices, the direction of the relationship, and the type of the relationship.

The graph DB **160** may be configured as a physical system, device, or component, as a virtual system, device, or component, or in a hybrid physical-virtual configuration. It may be understood that, while the graph DB **160** is depicted in FIG. **1**A as a discrete element external to the cloud environment **103**, the graph DB **160** may be included within any of the various elements of the network system **102**, including the cloud environment **103**, the various cloud platforms **104**, and subparts thereof, and the network **108**, without loss of generality or departure from the scope of the disclosure. Further, it may be understood that the graph DB **160** may be directly connected to, or realized as a component of, the graph analysis system **150**, without loss of generality or departure from the scope of the disclosure.

FIG. **1**B is an example diagram depicting a network system **100** and various associated network and external objects, according to an embodiment. The depicted network system **100** includes a cloud platform **110**, where the cloud platform **110** may be a cloud platform similar or identical to a cloud platform, **104**, of FIG. **1**A, above. The cloud platform **110** includes various subnets, **130-1** through **130-*n*** (hereinafter, "subnets" **130** or "subnet" **130**), and various security objects, **105-1** through **105-*m*** (hereinafter, "security objects" **105** or "security object" **105**). As is applicable to the subnets **130**, "n" is an integer having a value greater than or equal to two. Further, as applicable to the security objects **105**, "m" is an integer having a value greater than or equal to five. In addition, while the network system **100** of FIG. **1**B includes certain elements and combinations of elements, as well as connections therebetween, it may be understood that the depiction is provided for illustrative purposes, and that other, like, elements, combinations of elements, and connections therebetween may be implemented without loss of generality or departure from the scope of the disclosure. Other, like, network systems **100** may further include multiple cloud platforms **110**, including variously-interconnected cloud platforms **110**, and other, like, variations and configurations, without loss of generality or departure from the scope of the disclosure.

As described with respect to FIG. **1**A, above, the cloud platform **110** is a platform, architecture, or other, like, configuration providing for connectivity of the various systems, devices, and components described with respect to FIG. **1**B. The cloud platform **110** may be a commercially-available cloud system, provided on a service basis, such as, as examples and without limitation, Amazon AWS®, Microsoft Azure®, and the like. The cloud platform **110** may be a private cloud, a public cloud, a hybrid cloud, and the like. The cloud platform **110** may be implemented as a physical network of discrete, interconnected objects, and the like, a virtual network, providing for interconnection of various virtual systems and devices, as well as a hybrid physical-virtual network, including both physical and virtualized components. The cloud platform **110** may be, or may replicate or otherwise simulate or emulate, as examples, and without limitation, a local area network, a wide area network, the Internet, the World-Wide Web (WWW), and the like, as well as any combination thereof. Further, as described with respect to FIG. **1**A, above, the cloud platform **110** may include one or more subnets **130**, wherein each subnet **130** may be configured to serve as a cloud platform **110** for the various security objects **105** included in the subnet **130**, while retaining the connectivities and functionalities provided by the cloud platform **110**.

The cloud platform **110** may be configured to include an orchestrator **115**. The orchestrator **115** is configured to provide for management of the cloud platform **110**. The orchestrator **115** may be configured to provide one or more functionalities including, without limitation, monitoring of elements or components of the cloud platform **110**, logging and reporting data relating to the cloud platform **110**, managing cloud platform **110** updates and maintenance, generating cloud platform **110** alerts, as well as other, like, functionalities, and any combination thereof. The orchestrator **115** may be configured to report one or more data features related to the cloud platform **110**, such as may be requested during the execution of network analysis processes, such as those described hereinbelow.

The security objects **105** are security objects similar or identical to those security objects, **105**, of FIG. **1**A, above. As described with respect to FIG. **1**A, the security objects **105** are virtual entities or instances of systems, devices, or components, including virtual systems, devices, or components, or any combination thereof. Further, as described with respect to FIG. **1**A, security objects **105** may include, without limitation, security objects, network objects, and the like, as well as any combination thereof. Examples of security objects **105** include, without limitation, virtual networks, firewalls, network interface cards, proxies, gateways, containers, container management objects, virtual machines, subnets **130**, hubs, virtual private networks (VPNs), peering connections, load balancers, route tables, and the like, as well as any combination thereof.

External objects, as may be adjacent or relevant to a cloud platform **110**, are objects similar or identical to the security objects **105**. The external objects may be configured to communicate with one or more security objects **105**, with other, various, external objects, and the like, as well as any combination thereof.

FIG. **2** is an example flowchart **200** depicting a method for constructing security graphs, including network and computing objects, for cloud environments, according to an embodiment.

At S**210**, security objects are identified, and security object data is collected. In one embodiment, security objects may be identified by querying a cloud platform, through, for example, an orchestrator (e.g., orchestrator **115**, of FIG. **1**B, above), and the like. In an embodiment, S**210** may include submitting one or more requests to each cloud platform and collecting responses therefrom. The requests may include instructions directing the orchestrator to report information including, without limitation, the number of devices connected to or included in the cloud platform, the names of such devices, the types of such devices, other, like, information, and any combination thereof.

In an embodiment, identification of security objects and collection of security object data at S**210** includes querying each cloud platform, where such querying may include generation of one or more queries through an application programming interface (API), such as a REST API. Through the API, security objects' identities and description data are provided in response to such API queries. API queries may be pre-configured data requests, specified in the API, and configured to cause, for example, an orchestrator to return the one or more data features described herein. API queries may be generated based on one or more APIs, or the like, including generic APIs, such as REST, as well as platform-specific APIs, where such platform-specific APIs may be configured to provide for one or more predefined interactions with a cloud platform, such Amazon AWS®, Microsoft Azure®, and the like, where such predefined interactions may include, without limitation, security object identification and data collection.

Further, at S**210**, security object data is collected. Security object data is data describing one or more security objects, such as those objects, **105**, of FIG. **1**B, above. Security object data may include data describing, as examples and without limitation, object types, object names or unique identifiers (IDs), object network addresses, object port configurations, object status, such as online, offline, busy, and available, object input signal configurations, object output signal configurations, object security or access configurations, object processing rules and the like, as well as any combination thereof. Object data may be collected at S**210** from one or more sources including, without limitation, various networks, network monitors, subnets, external objects, security objects, and the like, such as the cloud platform, **110**, cyber-security system, **150**, subnets, **130**, external objects, security objects, **105**, cloud platform orchestrators, **115**, all of FIG. **1**B, above, and the like, as well as any combination thereof. Collection of security object data, at S**210**, may be executed via one or more means including, without limitation, generation and transmission of one or more API queries, such as are described hereinabove, by other, like, means, and any combination thereof.

As a first example, collection of security object data at S**210** may include collection of the identities of all objects included in a cloud platform by generation and transmission of an API query. In a second example, where a specific object, such as a given firewall, is specified in an API query, collection of security object data at S**210** may include collection of object data from a firewall, including the collection of firewall rules, collection of firewall event logs, collection of firewall port configurations, and the like, as well as any combination thereof. As a third example, collection of security object data at S**210** may include collection of object data from all virtual machines (VMs) in a cloud platform, where such VMs are described generally in an API query, including the collection of data resources or libraries internal to the VMs, VM port configurations, VM statuses, and the like, as well as any combination thereof.

At S**220**, a security graph is constructed. A security graph is a data feature describing the various objects included in, and adjacent to, a network, as well as the relationships between such objects. A security graph may be constructed based on data including, without limitation, data relevant to the objects identified at S**210**, from which data is collected, and the like, as well as any combination thereof. A security graph may be constructed in one or more various formats including, without limitation, a table, chart, or other, non-visual, data organization format, a list of objects, other, like, formats, and any combination thereof, where such formats may provide security object information including, without limitation, descriptions of security objects, properties, relations, and the like, as well as any combination thereof. In an embodiment, construction of a security graph, at S**220**, may include construction of a visual "node and link" graph. An example security graph schema, generated in a visual format and presented through a security graph utility, is described with respect to FIG. **3**A, below. An additional security graph data feature, including a list of security objects, where the list is configured to describe an object-to-object path, is described with respect to FIG. **3**B, below.

At S**230**, relationships between security objects are determined. Security object relationships are descriptions of the various connections between the security objects identified at S**210**. Network relationships may describe aspects of the connections between objects including, without limitation, connected objects, relevant ports of connected objects, connection bandwidths, connection durations, connection protocols, connection names or IDs, connection statuses, and the like, as well as any combination thereof.

In an embodiment, security object relationships may be determined at S**230** using a static analysis process. In this embodiment, the static analysis may include analysis of object and protocol code and rules, based on simulated network operation, as collected at S**210**, to provide for identification of security object relationships based on security object configurations. As an example of a static analytic determination, data collected from a firewall at S**210** may specify, in the firewall's port configurations, communication with a first device on a first port using a first protocol and connection with a second device on a second port using a second protocol. Further, according to the same example, the firewall object may include one or more instructions specifying transmission of a specific log file, via a third port, to a connected repository. According to the same example, security object relationships determined at S**230** may include connections between the firewall and the first device, connections between the firewall and the second device, and connections between the firewall and the repository.

Determination of security object relationships at S**230** may further include updating the graph or graphs constructed at S**220** to include the determined relationships. Graphs may be updated at S**230** by associating one or more data labels, tags, or other, like, features with a graph entry for a security object determined to have a relationship with another object. The association of data labels and tags may further include the association of labels or tags describing various aspects of the determined relationship or connection including, as examples and without limitation, connection source and destination, connection type, connection direction, connection status, connection protocol, and the like, as well as any combination thereof. Accordingly, as an example, determination, at S**230**, of a relationship between two objects may include the association of a data label or tag with each object included in the relationship, the data label or tag describing the same relationship for each object. Further, in an embodiment, where a graph is presented as a visual representation of a network system, such as a "node-and-link" graph, updating of the graph, at S**230**, based on determined relationships, may further include updating the visual graph to include visible "links" or connections between object "nodes," such as by, as examples and without limitation, updating the original visible graph to include such links, adding a second, visible overlay to the graph, including the links, and the like, as well as any combination thereof.

In addition, determination of security object relationships at S**230** may include analysis of such determined relationships to identify impermissible relationships. Where determination at S**230** includes such permissibility analysis, such analysis may include, without limitation, comparison of determined relationships with one or more dictionaries, or other, like, repositories of object relationship information, to determine whether a given relationship matches a predefined relationship included in the dictionary, where such a pre-defined relationship may be pre-tagged as "permissible," "not permissible," or the like. Where a determined relationship is determined to match a relationship which has been pre-defined as "not permissible" or as otherwise unacceptable, the relationship may be removed from the graph, such as by updating the graph in a manner similar to that described with respect to adding relationships to the graph, with the update providing for removal of one or more specified unacceptable relationships.

Further, in an embodiment, security object relationships may be determined at S**230** by application of observational or active logging methods, such as those methods providing for detection of object-to-object connections by monitoring traffic of a network in use.

At S**240**, the generated security graph is saved in a graph database, such as the graph DB **160**. The generated security graph includes any computing and/or network objects in the cloud environment of an organization. Such objects may be represented as nodes in the graph database, and their connections as edges. Each node, corresponding to each entity, stored in the database may include a recording describing its properties. Such properties may include "known network properties" (information that can be retrieved from querying the entity or the cloud platform) and calculated properties. One of the calculated properties includes a "reachable" or reachability property. The reachability property defines if and how an entity can be reached from an external or internal network. For example, an address, a port number, a protocol, and the like, or combination thereof, would describe how to reach the object. The various embodiments for reachable property are discussed in greater detail below.

It should be noted that, based on the known and calculated properties, security insights may be generated. Security insights are natural-language representations of aspects of the security graph constructed at S**220**. Security insights may include pure-text descriptions of objects and relationships. An example of a pure-text object relationship description may be "firewall one is connected to object two, which is a VM, and object three, which is a load balancer." Such representations may be in a query format.

In addition, security insights may include detailed descriptions of objects, relationships, and the like, as well as any combination thereof. An example of a detailed object relationship description may be "the gateway is currently active, and is connected to the VM via the second port, using the first protocol."

As another example, applicable to a multiple-step relationship, an insight may specify a path from "virtual machine one to load balancer five, where port eighty is routed to port **1337** on virtual machine one, then from load balancer five to firewall two, where port eighty of firewall two is open, then from firewall two to subnet sixteen, where subnet sixteen has a network address of 10.0.1.0/24, then from subnet sixteen virtual network ten, where virtual network ten has a network address of 10.0.0.0/16, then from virtual network ten to virtual network eleven, via peering connection twelve, where peering connection twelve includes a routing rule to virtual network twelve, specifying virtual network twelve's network address, where virtual network twelve's network address is 172.31.0.0/16." The insight described with respect to the second example may be interpreted to describe a virtual machine accessible from a virtual network via a series of hops, where only specific ports and addresses are allowed and routed through the firewall.

Further, generation of security insights at may include the generation of insights providing for, without limitation, description of network management or anomaly detection data, descriptions of network configurations or events which are rare or novel, such as connection of a new device to a network, descriptions of connections which are unauthorized, such as re-connection of a user's device to a subnet which the user is not permitted to access, descriptions of connections which display anomalous behavior, such as connections displaying spikes of network activity, and the like. Such insights may be generated according to one or more pre-defined or user-defined filters, rules, and the like, as well as any combination thereof. As an example, generation of security insights may include generation of an insight specifying that "VM thirty normally connects to load balancer twenty and firewall eight but is currently only connected to load balancer twenty."

In addition, insights may further include, without limitation, high-level insights, where high-level insights include information describing one or more features of a network which may not be detectable based on the analysis of individual objects. Examples of high-level insights may include, without limitation, "third-party networks A, B, and

C currently have access to the internal network," "objects E, a database, and F, an administrator interface, are currently exposed to external networks," and "cross-environment exposure has been detected between the development and production environments."

The security objects in the graph database may be tagged based on properties and insights. Tagging of security objects at may include, without limitation, association of one or more data labels, tags, or other, like, features, with graph entries for one or more security objects, including entries in graphs such as those described with respect to S**220**, above. The data labels, tags, or other, like, features, with graph entries, may include descriptions of aspects of relevant graphs, objects, relationships, and the like, as well as any combination thereof. Examples of relevant descriptions include, as examples and without limitation, insights, such as those generated at S**240**, object relationships, such as those determined at S**230**, object details, such as may be collected at S**210**, descriptions of whether an object appears in another graph, object type counts, per-object connection counts, graph connection counts, descriptions of an object's open ports and counts thereof, descriptions of an object's network address or addresses, descriptions of protocols relevant to an object, and other, like, descriptions, as well as any combination thereof.

FIG. **3**A is an example flowchart **300** depicting a method for determining reachable properties of security objects, according to an embodiment. A reachable property defines if and how an object on the generated security graph can be reached from an external or internal network, and/or an external or internal object. External means outside of the cloud environment of an organization. An object may be any computing or network object designated in a security graph generated as discussed above.

At S**305**, a security graph is accessed or otherwise obtained from the graph database. Within a security graph, various objects or entities, as may be included in a network or cloud environment of an organization, may be represented as "nodes" or "vertices," and such "nodes" or "vertices" may be interconnected by one or more "links" or "edges," the "links" or "edges" representing the relationships between the various objects included in a network or environment. Each object in the graph may be associated with known properties of the object. Examples for such properties may include an object's name, its IP address, various predefined security rules or access rules, and the like.

At S**310**, possible network paths within the obtained security graph are identified. A network path is a connection of two or more security objects accessible from an external or internal network, and/or an external or internal object. That is, a network path may include sequential representations of possible data/control flows between two or more objects in a graph. In an embodiment, where two objects in a graph are represented as vertices, and where the vertices are joined by an edge, a path may be constructed between the two vertices. A path may be a vertex-only path, describing a sequence of vertex-to-vertex "hops," an edge-only path, describing only the edges included in the sequence without description of the associated vertices, or a combined edge-vertex path, describing both edges and vertexes included in the sequence.

According to disclosed embodiments, a path shows a connection between security objects and/or computing objects that communicate over a network. An object may be a virtual, physical, or logical entity.

In an embodiment, paths can be identified by traversing the security graph. The traversal can start or end at objects that are connected to an external network (the internet). The traversal of the security graph can be performed using solutions disclosed in the related art, e.g., a breadth-first search (BFS), a tree traversal, and the like, as well as any combination thereof.

In another embodiment, paths can be identified by querying the graph database storing the security graph. Examples of applicable queries include, without limitation, queries configured to identify all paths between a first graph object (node) and a second graph object, queries configured to identify all paths between all graph vertices of a first object type and all graph vertices of a second object type, other, like, queries, and any combination thereof.

Following as performed at S**3150** through S**330**, the list of paths are iteratively identified to determine the reachability properties of the path. Specifically, at S**315**, a path list is populated to include all identified paths. A path list may be a table, list, or other type of data structure. A path list may be unordered or ordered, including ordering according to one or more path properties. At S**320**, a path from the path list is selected. At a first run of the method a first path in the list is selected. At S**325**, path elements are analyzed to determine reachable properties. Path element analysis, as at S**325**, is an iterative analysis of each element included in the path selected at S**320**. The operation of S**325** is discussed in detail with reference to FIG. **3**B.

At S**330**, it is determined whether the last path of the path list has been analyzed, and if so, execution terminates; otherwise, execution returns to S**320**.

FIG. **3**B is an example flowchart S**325** depicting the analysis of a network path to determine reachable properties of objects included in the path, according to an embodiment.

At S**355**, elements within a selected network path are identified. Elements are network and/or computing objects and relationships (or connections) between such objects. Identification of elements within the selected path may include, without limitation, identification based on properties, and other, like, data, included in the elements, identification of elements based on element identifications provided during the execution of S**310** of FIG. **3**A, above, and the like, as well as any combination thereof. Further, identification of in-path elements may include identification of element properties or attributes including, without limitation, names, network addresses, rulesets, port configurations, and the like, as well as any combination thereof.

Then, at S**360** through S**380**, the list of paths are iteratively processed in order to determine reachable properties of the elements. Specifically, at S**360**, the next element is selected. The next element is a subsequent element of the set of elements, within the selected path, identified at S**355**. Where execution of S**360** follows the execution of S**380**, the next element may be an element which, in the selected network path, immediately follows the element relevant to the preceding execution of S**370** and S**375**. Where execution of the method described with respect to FIG. **3**B includes a first execution of S**360**, the first execution of S**360** may include the selection of a first element of the selected path.

For exemplary purposes, a network path may be a path from a virtual machine (VM), connected to a NIC, connected to a load balancer, connected to a firewall. According to a first example, where S**360** is executed for the first time, the first execution of S**360** may include the selection of the VM as the selected element. Further, according to a second example, where execution of S**360** follows execution of S**380**, selection of a next element at S**360** may include selection of, following the VM, selection of the NIC, or, following the NIC, selection of the load balancer, or, following the load balancer, selection of the firewall.

At S**365**, it is determined whether the selected element has been analyzed. Determination of whether the selected element may include the determination of whether one or more reachable properties are included in the relevant graph element. As execution of S**375** provides for the population of reachable properties into the security graph, an element which does not include such reachable properties in the graph may be assumed to have not been analyzed.

Where, at S**365**, it is determined that the selected element has been analyzed, execution continues with S**360**. Where, at S**365**, it is determined that the selected element has not been analyzed, execution continues with S**370**.

At S**370**, reachable properties are determined. Reachable properties are object properties describing if, and how, a given path element is reachable through the selected path, and, specifically, from an external/internal network. Examples of reachable properties include, without limitation, binary properties describing whether an element is reachable, protocols by which the element is reachable, network addresses at which an element is reachable, ports by which an element is reachable, access rules, and the like, as well as any combination thereof.

In an embodiment, a reachable property is determined as a minimal set of reachable properties of all other objects in the path. As a simple example, if a path includes two objects, where one object can receive traffic from any source IP address through port **1515**, and the other object can receive traffic only from a source IP address of 173.54.189.188, the reachable property of the second object may be that the second object is reachable through "source IP address 173.54.189.188 and port **1515**."

An example demonstrating the operation of S**370** is discussed in detail with reference to FIG. **5**.

At S**375**, reachable properties are populated into the security graph. Reachable properties, as may be determined at S**370**, may be populated into the graph by processes including, without limitation, labeling or tagging graph vertices, updating network or graph object properties, generating one or more graph overviews, layers, or graph-adjacent data features, and the like, as well as any combination thereof.

In an embodiment, population of reachable properties into the security graph may include, for each object, population of object network access control lists (NACLs) as described hereinbelow, into the security graph elements corresponding with the various path elements, as well as the population of scope specific NACLs, and other, like, properties into the graph. Scope-specific NACLs are NACLs describing object, path, or network accessibility properties specific to a given scope, where a given scope may be the internet, various given accounts, various given environments, and the like. Scope-specific NACLs may, for example, describe the properties of an object with respect to the object's internet accessibility, where the object may be configured to include different access control properties for internet access and local intranet access.

Further, population of reachable properties into the graph may include population of one or more paths into the graph, including by population processes similar or identical to those described with respect to population of individual objects. Population of paths into the graph may include, without limitation, population of one or more paths into the graph, including a presently-analyzed path, population of one or more path properties, and the like, as well as any combination thereof. Path properties, as may be populated to a graph, are properties describing various attributes of a path, including, without limitation, NACLs applicable to path elements, path segments, or full paths, including full-path aggregate NACLs, and the like, as well as any combination thereof. Further, population of path properties into the graph may include the population of one or more scope-specific path properties, where such scope-specific path properties may be properties relevant to specific scopes, such as those described herein.

Where population of reachable properties includes labeling or tagging a graph, or elements thereof, one or more graph vertices or edges, the corresponding objects or relationships, or both, may be labeled, tagged, or otherwise associated with one or more data features describing relevant reachable properties. In addition, where population of reachable properties to the graph includes updating graph objects, graph vertices and edges, the corresponding objects and relationships, or both, may be directly updated to explicitly include the calculated properties.

Further, where population of reachable properties includes the generation of one or more graph layers or overlays, the generated graph layers or overlays may be data features independent of, but corresponding to, the relevant graphs, where the generated overlays or layers may include one or more data features describing the reachable properties of the various graph elements.

At S**380**, it is determined whether all elements in the selected path have been analyzed. Determination of whether all elements in the selected path have been analyzed may include, without limitation, determination of whether the immediately preceding execution of S**375** relates to the last element in the selected path, determination of whether additional elements remain in the path, determination of whether any additional in-path elements have been analyzed, and the like, as well as any combination thereof.

Where, at S**380**, it is determined that all elements in the selected path have not been analyzed, execution continues with S**360**. Where, at S**380**, it is determined that all elements in the selected path have been analyzed, execution terminates.

FIG. **4**A is an example screenshot of a security graph visualization **400**, generated according to an embodiment. The example security graph visualization **400** is generated as a visual representation of a network and is presented through a security graph utility. A security graph utility may be an application, interface, or other, like, means of providing a visual representation of a security graph, and the like, where the provided security graph visualization **400** may include various interactive features, as described hereinbelow. A security graph utility may be configured as, as examples and without limitation, a web interface, an application or executable installed on a user or administrator device, other, like, configurations, and any combination thereof.

The security graph visualization **400** of FIG. **4**A is a security graph visualization representing a network, such as the networks described hereinabove, wherein the various objects, systems, devices, components, and the like, of the network are represented as vertex visualizations **410**, wherein such vertex visualizations are variously interconnected by edge visualizations **420**, representing connections between the various objects, systems, devices, components, and the like. It may be understood that while only one vertex visualization **410** and one edge visualization **420** are labeled for purposes of simplicity, other, like, vertex visualizations **410** and edge visualizations **420** may be so labeled without loss of generality or departure from the scope of the disclosure.

The security graph visualization **400**, and corresponding security graph utility, may be configured to provide for various interactive functionalities. In an embodiment, where a user interacts with a node vertex visualization **410**, such as by clicking the vertex visualization **410** with a mouse or tapping the vertex visualization **410** through a touchscreen, the graph utility may be configured to display a vertex overview pane **415**. The vertex overview pane **415** may be an information panel, including data relating to the given vertex visualization **410** and describing various object data features, such as those object data features identified or determined during the execution of a process similar or identical to that described with respect to FIG. **2**. The vertex overview pane **415** may be configured to provide information relating to the various vertex visualizations **410** including, as examples and without limitation, object names, types, statuses, relevant metadata, and the like, as well as any combination thereof.

Further, the security graph visualization **400**, and corresponding security graph utility, may be configured to include a search tool **430**, providing for location and selection of one or more user-specified vertex visualizations **410** or edge visualizations **420** within the graph. The search tool **430** may be configured to provide for search functionality based on one or more user specifications including, as examples and without limitation, object names, types, IDs, statuses, labels or tags associated with various elements of the security graph visualization **400**, and the like, as well as any combination thereof. In addition, the security graph visualization **400**, and corresponding security graph utility, may be configured to include a help tool **440**, providing for display of one or more resources related to the security graph visualization **400** and security graph utility.

It should be noted that a security graph visualization **400**, shown in FIG. **4**A, may be constructed in other formats including, without limitation, a table, chart, or other, non-visual, data organization formats, a list of objects, other, like, formats, and any combination thereof.

FIG. **4**B is an example security graph object list **450**, configured to provide information describing object-to-object routing within a security graph, according to an embodiment. The security graph object list **450** includes a mode indicator **460**, a mode-specific data display **465**, and a list of objects, **470-1** through **470**-*n* (hereinafter, "object" **470** or "objects" **470**), where 'n' is an integer having a value greater than or equal to two. It may be understood that, while the provided security graph object list is configured to provide a list of objects **470** arranged according to a defined mode, other, like, modes and object **470** arrangements may be similarly applicable without loss of generality or departure from the scope of the disclosure.

The security graph object list **450** is a list of objects within a network, a segment of a network, a path of a network, and the like, as well as any combination thereof. A security graph object list **450** may be generated or provided as a function of one or more methods including, without limitation, those methods described herein, other, like, methods, and any combination thereof. A security graph object list **450** may be, without limitation, a feature of a security graph management tool or utility, such as a tool or utility configured to provide the security graph visualization of FIG. **4**A, above, a stand-alone tool or utility, and the like, as well as any combination thereof. The security graph object list **450** may be configured to list security objects **470** in one or more orders based on factors including, without limitation, object names, object types, object connection latencies, mode-specific factors, other, like, factors, and any combination thereof. Where the security graph object list **450** is configured to list security objects **470** based on mode-specific factors, the mode indicator **460** may be configured to provide information describing a specific list mode, and the mode-specific data display **465** may be configured to provide information describing the contents of the list **450** in relation to one or more selected modes.

Modes are selected list-organization profiles, configured to provide for population of a security graph object list **450** in one or more configurations. Modes may provide for configuration of a security graph object list **450**, including configurations specific to, as examples and without limitation, routing paths, object utilization or availability descriptions, other, like, configurations, and any combination thereof. Where a given mode is selected, the selected mode may be displayed via a mode indicator **460**, where the mode indicator is configured to provide descriptive information regarding a selected mode. Further, where a mode is selected, the mode-specific data display **465** may be configured to display information regarding a specific security graph object list **350** populated based on the specified mode or modes.

As an example, with reference to the provided FIG. **4**B, a "route" mode may be selected, providing for population of a security graph object list **450** with objects occupying a data path between a first object **470-1** and a destination object **470**-*n*. According to the same example, the mode indicator **460** may be configured to display "Route," indicating that the security graph object list **450** is populated to provide information describing a route between the specified objects, and the mode-specific data display **465** may be configured to display "9 hops," indicating that a transmission from a first object **470-1** makes nine "hops," or transmissions between objects, before reaching the destination object **470**-*n*. Further, according to the same example, the security graph object list **450** may be configured to include and display each security object **470** through which the transmission passes, including the first object **470-1** and the destination object **470**-*n*, in the order of transmission.

FIG. **5** is an example path illustration **500** that includes a database **510**, connected to a network interface (NIC) **520**, connected to a public proxy **540**, where a security group **530** is applied on the NIC **520**.

The database **510**, provided as the first element in the given path, includes a private IP address value, included in the element's properties, of 10.0.0.1, providing for accessibility of the database **510** only to traffic with a destination address of 10.0.0.1. The network interface **520** includes a port configuration specifying port **1234**, included in the element's properties, providing for configuration of the network interface **520** to receive traffic through port **1234**. The security group **530** includes a filtering rule, included in the element's properties, specifying source addresses 2.4.6.8 and 1.3.5.7, providing for transmission only of traffic originating from address 2.4.6.8 or address 1.3.5.7. The public proxy **540** includes a translation rule, included in the element's properties, which specifies translation of frontend port **1234** to backend port **4321**.

To determine the reachable properties of the database **510**, or the properties of another, like, element, the traffic permitted to pass through each object-to-object step of the path is determined based on the properties of each object. Where each object in the path includes properties describing permissible or impermissible transmissions or connections, such properties may be represented as network access control lists (NACLs). An NACL, according to the provided example, is a data feature describing one or more properties of allowed traffic. Further, in an embodiment, an NACL may be a data feature describing one or more properties of impermissible traffic. An NACL, according to the example, may be structured according to the examples shown below.

[port=from x11 to x12, IP=from y11 to y12, . . .]

[port=from x21 to x22, IP=from y21 to y22, . . .]

The provided example NACLs describe a set of transmission or connections which are permitted to pass through a network object having such NACLs. The provided example NACLs include a first NACL which provides for the allowance of connections or transmissions over ports x11 through x12 and with source IPs between y11 and y12, and a second NACL which provides for the allowance of connections or transmissions over ports x11 to x12 with source IPs between y11 and y22. Further, according to the same example, all traffic matching either of the provided NACLs may be considered allowed, while all other traffic may be considered blocked.

During the analysis of each path object, the aggregated allowed traffic, or all traffic which matches the NACLs of each preceding object, may be further restricted by comparing the aggregate allowed traffic with the NACLs of the analyzed path object. Such a restriction may be described as an intersection set operation, whereby the set of the aggregate allowed traffic, the set including all traffic not restricted by the NACLs of preceding path objects, may be compared with the set of all traffic permitted by the NACLs of the analyzed path object to identify a set of traffic which is both included in the aggregate allowed traffic and allowed by the analyzed path object's NACLs. The resulting common set of traffic may be, for analysis of subsequent path objects, considered as the set of allowed traffic, providing for the execution of similar analyses for all subsequent path objects.

As a further example, providing a high-detail description of a path analysis similar to that described above, a known network path may be from a database object **510**, to a network interface object **520**, to a security group object **530**, to a public proxy object **540**. According to the same example, the path analysis of the described path may be an analysis of "all possible traffic." Beginning with the database object **510**, path analysis may include determination, based on properties of the database, that the database object **510** has a private IP address of 10.0.0.1. Next, proceeding to the network interface object **520**, path analysis may include, based on the network interface object's **520** collected properties, determination that the only network interface object **520** port relevant to database object traffic is port number **1234**, providing for the restriction of the initial "all possible traffic" to "all traffic with a destination IP address of 10.0.0.1, a port number of 1234, and any source." Subsequently, proceeding to the security group object **530**, path analysis may include, based on the security group's **530** collected properties, determination that the security group object **530** only allows traffic from sources 2.4.6.8 and 1.3.5.7.

Following the determination of the security group object's **530** rules, the previously-restricted "all possible traffic" may be further restricted to "all traffic with a destination IP address of 10.0.0.1, a port number of 1234, and a source of 1.3.5.7," and "all traffic with a destination IP address of 10.0.0.1, a port number of 1234, and a source of 2.4.6.8." Further, proceeding to the public proxy object **540**, path analysis may include determination that, based on the collected public proxy object **540** properties, the public proxy object **540** includes a rule which translates frontend port **4321** to backend port **1234**. Accordingly, based on the identified public proxy object **540** rule, the already-restricted "all possible traffic" may be further restricted to "all traffic with a destination IP address of 10.0.0.1, a port number of 4321, and a source of 1.3.5.7," and "all traffic with a destination IP address of 10.0.0.1, a port number of 4321, and a source of 2.4.6.8." Subsequently, all allowable traffic which matches the specified restriction, as well as allowable traffic from other paths, may be associated with the graph element representing the database object **540**, providing for determination of additional reachable properties for the database object **540**, such as, as examples, "numAddressesOpenToNonStandardPorts=2," "numPortsOpenToInternet=1," and accessibleFromInternet=true."

In an embodiment, determination of reachable properties by path analysis may include execution of one or more traffic set operations similar to those described in the provided examples. Where determination includes such execution, relevant traffic set operations may include, without limitation, finding the intersection of two sets, finding the union of two sets, subtracting one set from another, finding the complement of a set, and the like, as well as any combination thereof. Further, in an embodiment, determination of reachable properties may include, for each step of a security graph path, one or more high-level operations including, without limitation, firewall operations, proxy and network address translation (NAT) operations, such as making address translations, gateway, virtual network, and subnet operations, such as checking routing rules to determine whether a routing rule relevant to an internet gateway is relevant to path traffic, checking routing rules, checking security rules, checking network paths and the like, as well as any combination thereof. The described high-level operations may, further, include the execution of one or more traffic set operations, such as those described hereinabove.

In an embodiment, reachable properties may, for various scopes, such as the internet, other virtual networks, other accounts, other environments, and the like, include, without limitation, whether a virtual machine (VM) or resource is accessible from a given scope, the number of addresses open to a given scope, the number of ports open to a given scope, possible paths, including paths formatted as lists of objects, traffic allowed in each path, and the like, as well as any combination thereof.

In addition, in an embodiment, determination of reachable properties may further include the creation of one or more edges on a graph, based on the results of the determinations. As an example, where a VM is determined to be accessible from a specific object, an edge may be added to the graph to represent the relationship between the VM and the specific object. Such creation of edges may provide for subsequent integration of the described relationships into a graph search.

Such determination of reachable properties, and values thereof, may provide for, in an embodiment, a separate "reachable properties" layer of a graph. Such a separate layer may be configured to provide for various functionalities including, without limitation, identification of one or more reachable properties for various security objects, generation of path-based or object-based insights, execution of queries over a graph, and the like, as well as any combination thereof.

FIG. **6** is an example hardware block diagram **600** depicting a cyber-security system **150**, according to an embodiment. The cyber-security system **150** includes a processing circuitry **610** coupled to a memory **620**, a storage **630**, and a network interface **640**. In an embodiment, the components of the cyber-security system **150** may be communicatively connected via a bus **650**.

The processing circuitry **610** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), graphics processing units (GPUs), tensor processing units (TPUs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

The memory **620** may be volatile (e.g., random access memory, etc.), non-volatile (e.g., read only memory, flash memory, etc.), or a combination thereof.

In one configuration, software for implementing one or more embodiments disclosed herein may be stored in the storage **630**. In another configuration, the memory **620** is configured to store such software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the processing circuitry **610**, cause the processing circuitry **610** to perform the various processes described herein.

The storage **630** may be magnetic storage, optical storage, and the like, and may be realized, for example, as flash memory or another memory technology, compact disk-read only memory (CD-ROM), Digital Versatile Disks (DVDs), or any other medium which can be used to store the desired information.

The network interface **640** allows the cyber-security system **150** to communicate with the various components, devices, and systems described herein for generation of network analysis insights, as well as other, like, purposes.

It should be understood that the embodiments described herein are not limited to the specific architecture illustrated in FIG. **6**, and other architectures may be equally used without departing from the scope of the disclosed embodiments.

It should be noted that the computer-readable instructions may be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code, such as in source code format, binary code format, executable code format, or any other suitable format of code. The instructions, when executed by the circuitry, cause the circuitry to perform the various processes described herein.

The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units (CPUs), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform, such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

As used herein, the phrase "at least one of" followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including "at least one of A, B, and C," the system can include A alone; B alone; C alone; A and B in combination; B and C in combination; A and C in combination; or A, B, and C in combination.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

What is claimed is:

1. A method determining reachability properties of security objects, comprising:

accessing a security graph, wherein the security graph lists all security objects and corresponding connections of the security objects in a cloud environment of an organization;

identifying a plurality of network paths in the cloud environment, wherein each network path includes at least two security objects accessible in the cloud environment;

for each of the plurality of identified network paths, iteratively analyzing each security object in a respective network path to determine its reachability properties, wherein the reachability properties of a security object as a minimal set of reachable properties of all other security objects in the respective network path, wherein the reachability properties of a security object define if and how the security object is reachable through the respective network path from at least a network external to the cloud environment; and

populating the security graph with the determined reachability properties of each security object.

2. The method of claim **1**, wherein identifying a plurality of network paths further comprises:

traversing the at least a security graph.

3. The method of claim **1**, wherein the security graph is saved in a graph database, and wherein identifying a plurality of network paths further comprises:

querying the graph database to retrieve the network paths.

4. The method of claim **1**, wherein analyzing a security object in a respective network path further comprises:

determining network properties defined for each security objects in the security graph; and

restricting the network properties of the security object based on the aggerated allowed traffic matching reachability properties of each preceding security object in the network path, wherein the restricted network properties of the security object are the reachability properties of the security objects.

5. The method of claim **4**, further comprising:

representing reachability properties of a security object as a network access control list (NACL).

6. The method of claim **5**, further comprising:

performing at least one operation on two NACLs of two security objects to determine reachability properties, wherein the at least one operation is any one of: union, subtraction, complement, and intersection.

7. The method of claim **1**, wherein the reachability properties include any one of: a source IP address, a destination IP address, a port number, and a security group.

8. The method of claim **1**, wherein populating the security graph with the determined reachability properties of each security object further comprises:

associating each security object in the graph with the security object's determined reachability properties and respective security graph.

9. The method of claim **1**, wherein the analysis of each security object in a respective network path is a static analysis, thereby the static analysis does not require simulating of network traffic.

10. A non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to execute a process for determining reachability properties of security objects, the process comprising:

accessing a security graph, wherein the security graph lists all security objects and corresponding connections of the security objects in a cloud environment of an organization;

identifying a plurality of network paths in the cloud environment, wherein each network path includes at least two security objects accessible in the cloud environment;

for each of the plurality of identified network paths, iteratively analyzing each security object in a respective network path to determine its reachability properties, wherein the reachability properties of a security object as a minimal set of reachable properties of all other security objects in the respective network path, wherein the reachability properties of a security object define if and how the security object is reachable through the respective network path from at least a network external to the cloud environment; and

populating the security graph with the determined reachability properties of each security object.

11. A system for for determining reachability properties of security objects, comprising:

a processing circuitry; and

a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to:

access a security graph, wherein the security graph lists all security objects and corresponding connections of the security objects in a cloud environment of an organization;

identify a plurality of network paths in the cloud environment, wherein each network path includes at least two security objects accessible in the cloud environment;

for each of the plurality of identified network paths, iteratively analyzing each security object in a respective network path to determine its reachability properties, wherein the reachability properties of a security object as a minimal set of reachability properties of all other security objects in the respective network path, wherein the reachability properties of a security object define if and how the security object is reachable through the respective network path from at least a network external to the cloud environment; and

populate the security graph with the determined reachability properties of each security object.

12. The system of claim **11**, wherein the system is further configured to:

traverse the at least a security graph.

13. The system of claim **11**, wherein the security graph is saved in a graph database, and wherein the system is further configured to:

query the graph database to retrieve the network paths.

14. The system of claim **11**, wherein the system is further configured to:

determine network properties defined for each security objects in the security graph; and

restrict the network properties of the security object based on the aggerated allowed traffic matching reachability properties of each preceding security object in the network path, wherein the restricted network properties of the security object are the reachability properties of the security objects.

15. The system of claim **14**, wherein the system is further configured to:

represent reachability properties of a security object as a network access control list (NACL).

16. The system of claim **14**, wherein the system is further configured to:

perform at least one operation on two NACLs of two security objects to determine reachability properties, wherein the at least one operation is any one of: union, subtraction, complement, and intersection.

17. The system of claim **11**, wherein the reachable properties include any one of: a source IP address, a destination IP address, a port number, and a security group.

18. The system of claim **11**, wherein the system is further configured to:

associate each security object in the graph with the security object's determined reachability properties and respective security graph.

19. The system of claim **11**, wherein the analysis of each security object in a respective network path is a static analysis, thereby the static analysis does not require simulating of network traffic.

* * * * *